UNIVERSITY OF COPENHAGEN
FACULTY OF SCIENCE

PhD Thesis

Christian Hansen

Sequential Modelling with Applications to Music Recommendation, Fact-Checking, and Speed Reading

Advisors: Stephen Alstrup, Christina Lioma, Jakob Grue Simonsen

Handed in: April 30, 2021
This thesis has been submitted to the PhD School of The Faculty of Science, University of Copenhagen

Abstract

Sequential modelling entails making sense of sequential data, which naturally occurs in a wide array of domains. One example is systems that interact with users, log user actions and behaviour, and make recommendations of items of potential interest to users on the basis of their previous interactions. In such cases, the sequential order of user interactions is often indicative of what the user is interested in next. Similarly, for systems that automatically infer the semantics of text, capturing the sequential order of words in a sentence is essential, as even a slight re-ordering could significantly alter its original meaning. This thesis makes methodological contributions and new investigations of sequential modelling for the specific application areas of systems that recommend music tracks to listeners and systems that process text semantics in order to automatically fact-check claims, or ”speed read” text for efficient further classification.

For music recommendation, we make three contributions: Firstly, a study of how the complexity of sequential music recommender methods relates to the diversity and relevance of the recommendations, and how diversification of recommendations can be used to control this trade-off. Secondly, we investigate how listening context impacts music consumption, which we use to motivate a new way of representing user profiles that captures sequential and contextual deviations from the user’s typical music preferences. Thirdly, we improve the prediction of music skip behaviour in a listening session based on past skips.

For fact-checking, we make three contributions: Firstly, we construct the currently largest benchmark dataset of naturally occurring claims for training automatic fact-checking models. Secondly, we link and use eye-tracking data of humans reading news headlines to automatic fact-checking predictions. Thirdly, we present two models for detecting check-worthy sentences for fact-checking, which by the use of weak supervision and contrastive ranking, make steps towards better model generalization in a domain with very limited training data.

Lastly, for speed reading, we contribute a new model that utilizes the inherent punctuation structure of text for learning how to ignore a large number of words, while being equally or more effective than processing every word in the text.

Dansk Resumé

Sekventiel modellering indebærer at skabe mening i sekventielle data, som naturligt forekommer i en bred vifte af domæner. Et eksempel er systemer, der interagerer med brugerne, logger brugerhandlinger og adfærd, og giver anbefalinger af potentiel interesse for brugerne på baggrund af deres tidligere interaktioner. I sådanne tilfælde er den sekventielle rækkefølge af brugerinteraktionerne ofte en indikation af, hvad brugeren i fremtiden er interesseret i. Tilsvarende er det vigtigt for systemer, der automatisk udleder semantikken i tekst, at repræsentere rækkefølgen af ord i en sætning, da selv en lille omstilling kan ændre dens oprindelige betydning væsentligt. Denne afhandling yder metodiske bidrag og nye undersøgelser af sekventiel modellering til specifikke anvendelsesområder for systemer der anbefaler musiknumre til lyttere, og systemer der behandler tekstsemantik for automatisk at foretage faktakontrol af udsagn eller ”speed-read” tekst for effektiv klassificering.

Til musikanbefaling yder vi tre bidrag: For det første en undersøgelse af, hvordan kompleksiteten af sekventielle musikanbefalingsmetoder er relateret til diversitet og relevansen af anbefalingerne, og hvordan diversificering af anbefalinger kan bruges til at kontrollere dette trade-off. For det andet undersøger vi, hvordan lyttekontekst påvirker musikforbruget, som vi bruger til at motivere en ny måde at repræsentere brugerprofiler, der fanger sekventielle og kontekstuelle afvigelser fra brugerens typiske musikpræferencer. For det tredje forbedrer vi forudsigelsen af hvor brugerer skipper musiknumre i en session baseret på tidligere skip.

Til faktakontrol yder vi tre bidrag: For det første konstruerer vi det i øjeblikket største benchmarkdatasæt med naturligt forekommende udsagn til træning af automatiske faktakontrolmodeller. For det andet forbinder vi og bruger eyetracking data fra mennesker, der læser nyhedsoverskrifter til automatiske faktakontrolforudsigelser. For det tredje præsenterer vi to modeller til detektering af kontrolværdige sætninger til faktakontrol, som ved hjælp af weak supervision og kontrastiv ranking tager skridt mod bedre model generalisering i et domæne med meget begrænsede træningsdata.

Endelig bidrager vi til speed reading problemet med en ny model, der udnytter tekstens tegnsætningsstruktur til at lære at ignorere et stort antal ord, samtidig med at den er mindst lige så god som hvis hvert ord i teksten bliver processeret.

Acknowledgements

First and foremost, I am grateful to my three supervisors, Christina, Jakob, and Stephen, who have provided me with many lessons on both academia and science as a whole, as well as personal lessons I will keep forever. My weekly meetings with Christina and Jakob have always been a source of joy, as they were both scientifically enlightening but also simply entertaining. I would also like to thank my fellow PhD students, with whom I have worked during the years, Lucas, Dongsheng, Stephan, and Niklas.

During my PhD, I have worked together with great people at Edulab to improve how mathematics is taught in Denmark. This has been an excellent collaboration where real-world problems could meet academia with exciting results. Of all the people at Edulab, I would especially like to thank Lotte, Kasper, Morten, Jakob, Stani, and Daniel.

I was fortunate to spend a summer away from Denmark as an intern at Spotify Research in London. I got to work with many wonderful people, where I would especially like to thank Rishabh, Brian, Federico, Lucas, and Mounia.

I also want to thank Rastin and Benjamin for a great collaboration at Danmarks Nationalbank. I realized how lucky I am that papers in computer science are so much shorter than in economics.

From my personal life, I would first like to thank all my friends and family who have put up with my wacky work schedule during the last few years. Especially I would like to thank Mikkel, Helena, and David, who have been a great support. Lastly, I would like to thank the two most important people during my PhD, my twin brother Casper and my fiancée Spring. Casper has been a great source of inspiration in our work together and has been the best collaborator anyone could ever hope to work with. Spring has been a constant source of support and love, who not only accepts that I like to talk about my work, but also perfectly feigns interest in it.

Contents

Chapter 1

Introduction

Many domains have a natural sequential ordering of their data, where ignoring the sequential ordering risks throwing away valuable information. One such domain is that of recommender systems, where a user's current preferences may be related to both recent interactions, characterising their current needs, and also long term behavior, expressing general preference patterns. In both the long and short term, how users interact with items happens in a sequential order, which typically implies a sequential dependency between the interactions. Examples of such sequential dependencies could be in e-commerce systems, where a user might be looking for clothes matching another recent purchase, or on music streaming platforms, where a user's current preference may be learned over a larger number of previous interactions. A seemingly different, but in fact highly analogous, domain is that of semantic inference of text, where the sequential word order is paramount for understanding meaning correctly, e.g., "*The lion ate a chicken*" or "*The chicken ate a lion*". This is broadly known as *term dependence* and has been long investigated by both linguists and computer scientists [70].

Utilizing and learning from sequential data is the topic of sequential modelling. In this thesis, we focus on music recommendation and textual fact-checking as two specific application areas of sequential modelling. We describe these below.

Music Recommendation

Recommender systems are integral in helping users navigate through the potentially huge number of items available on a content platform. They do so by presenting users with relevant items that match their preferences. Training such preference models is a challenge as the available training data is very sparse. This happens because a single user normally only interacts with a small fraction of the available items [25]. In this setup, two distinct ideas are used for learning user preferences: Firstly, similar users tend to have similar preferences, which can guide recommendation. Secondly, a user's preferences may be based on certain item properties, such that users can be recommended items similar to what they have previously enjoyed. These are the core ideas behind collaborative filtering [53] and content-based filtering [66, 81], respectively. To combine the benefits of both, many modern recommender systems are hybrid approaches utilizing both the item features and shared preferences [11, 20, 71].

Music recommendation has certain particularities different from the recommendation of other items, such as movies or books [74]. Consumption of music is highly sequential, as a single session easily consists of listening to tens of tracks, where the likelihood of a user enjoying a track is not only dependent on the track itself, but also on how it appears in the sequence [74]. Users also tend to repeat previously listened tracks [4], and have their current music preference strongly influenced by situational and contextual aspects [17, 51, 67]. Additionally, music listening is often a passive activity where the tracks are recommended and played automatically. This means that users have to actively skip a track to elicit a negative feedback signal. In contrast, not skipping a track cannot necessarily be interpreted as a positive feedback signal, as the user may simply not be actively engaged.

In this thesis, we incorporate and exploit the above mentioned particularities to design new methods for recommender systems. Firstly, we investigate how the complexity of the ranker used by the recommender system is related to both the relevance and diversity of the recommendations, where both sequential and non-sequential models are considered. To increase recommendation diversity, different diversification methods are investigated to explore their relevance and diversity trade-offs. Secondly, we perform a large-scale study on how context impacts music consumption, which we use to motivate a novel approach for generating dynamic user embeddings capturing sequential and contextual deviations from the user's typical music preferences. Thirdly, we investigate to what extent a user's skip behaviour can be predicted as a sequential classification task, where the first half of a session is used to predict the skip behavior of the second half, with the aim of better understanding the difficulties of modelling skips.

Fact-Checking

Misinformation is spreading at increasing rates, where especially fake news has a tendency to reach a larger audience and spread faster compared to news that is factually true [82, 93]. This spread of misinformation has long been considered a highly pressing societal issue by the World Economic Forum [47], and due to the scale of the issue, automatic solutions for fact-checking are necessary. An automated fact-checking pipeline [79] normally consists of three steps: (i) selecting check-worthy sentences, which are sentences containing a claim worth fact-checking, (ii) gathering related evidential information to those sentences which can help decide factuality, and (iii) using the evidence to infer the factuality of each check-worthy sentence. We describe each step below.

For step (i), the automatic selection of check-worthy sentences is considered a ranking task, which for a given text, e.g., transcribed political speeches and debates, aims to provide a ranked list of sentences in the order of how relevant they are to fact-check. A sentence is said to be check-worthy if it contains a factual claim that is of interest to determine the factuality of [46]. This means that even though a sentence like "*My dog is brown*" is factual, it is not check-worthy as its factuality would be of little interest to most people. Step (ii) and (iii) are typically considered jointly, as the modelling approach is determined by the structure of the input. Multiple different sources for obtaining evidential information have been considered for determining the factuality of a claim. This includes querying search engines with the claim as a query [52], or limiting the search to specific domains such as social media posts on Twitter [8]. Knowledge graphs have also been used for extracting facts related to the central entities of a claim [19]. Lastly, previously fact-checked claims can also be used as evidence when fact-checking new claims [76], as semantically similar claims may share the same factuality.

In this thesis, we make a series of contributions toward all three steps of the fact-checking pipeline. Firstly, we construct the largest dataset of naturally occurring claims. We do this by crawling claims from 26 fact-checking websites, where associated evidence obtained from a search engine and rich metadata is made available. We verify the dataset's usefulness for fact-checking by ablating the effectiveness improvement of including both the evidence and metadata. Secondly, we explore other modalities of fact-checking evidence, where we study to what extent eye-tracked data from users can be used for inferring factuality. Thirdly, we propose new check-worthiness models utilizing weak supervision and contrastive ranking to make more accurate predictions. Lastly, unrelated to the fact-checking pipeline, but related to the sequential modelling used in our proposed models, we consider the task of speed reading. Speed reading is the task of processing as few sequential inputs as possible without compromising model effectiveness. We propose a new model, which utilizes the inherent punctuation structure of text for learning how to ignore significant parts of the input sequence, while being equally or more effective than processing the entire sequence.

1.1 Research Outline

This thesis is composed of eight published articles, each of them presented as a separate chapter. These eight chapters are clustered into three themes, according to their domain of application. These three themes are: music recommendation, fact-checking, and speed reading.

This section provides an overview of the primary research questions tackled in each thesis chapter, and how these were investigated. For each research question, the relevant background material and main findings are briefly covered.

1.1.1 Music Recommendation

Chapter 2: Shifting Consumption towards Diverse Content on Music Streaming Platforms

The meaning of diversity in recommendations is influenced by the domain and task [54], but a general definition is that diversity is the opposite of similarity among the recommended items [12]. The aim of this work is to explore how diversity can be included in sequential music recommendation, where the user is passively listening or may choose to actively skip a given recommendation. The explicit active choice of skipping can be seen as interrupting the user experience, and is detrimental to the overall user satisfaction. Thus, when considering diversity in sequential recommendation, any recommendation given as a consequence of increasing the diversity should still be relevant to the user. This contrasts the work on list recommendation [6, 10, 72, 84], where the user is presented a number of items. In this case, if a subset of the recommended items are highly relevant, it is possible to include a selection of less relevant but more diverse items in the rest of the list.

Although there is a potential detrimental cost of increasing the diversity in sequential recommendation, there are a number of potential benefits which make diversity worth pursuing. The first one is that diversity can help users discover new interests [55, 90, 92], which have been linked to long term user retention [5, 64]. The second one is that diversity can help avoid the rich-get-richer problem [73], where a small subset of items receive a large amount of the interest as a consequence of how the recommender systems are trained.

The above leads to the following research questions:

RQ1 To what extent can diversity be included in sequential recommendation, and how can its effect on the relevance of the recommendations be controlled?

RQ2 What is the relation between the complexity of the ranker used by the recommender system and the diversity of the recommendations?

To answer **RQ1**, we first define two notions of diversity relevant to music recommendation. The first notion defines diversity with respect to popularity, and the second with respect to personalization. We investigate four different methods for increasing diversity, and we empirically evaluate how they affect the relevance of the recommended items to the user.

To answer **RQ2**, we evaluate four rankers of increasing complexity, where complexity of the ranker refers to the amount of user information and size of the model it uses for the recommendation. Specifically, we evaluate how each ranker fares with regards to both the effectiveness and diversity of the recommendations.

Our findings regarding both **RQ1** and **RQ2** show that i) it is possible to increase the diversity of the recommended items with little to no decrease in relevance, while even higher diversity can be achieved if a larger decrease in relevance is accepted; and ii) as the complexity of the rankers increases, there is a tendency for the recommendations to get more relevant but less diverse.

Chapter 3: Contextual and Sequential User Embeddings for Large-Scale Music Recommendation

In large-scale settings, it can be beneficial from an efficiency perspective to express user-item relevance using a simple vector operation between a user and item embedding [9], such as the cosine similarity. This contrasts the methods used for answering **RQ1-2**, where the relevance between a user and an item was estimated by more computationally expensive models. However, embedding-based methods are not limited to using a static user embedding, which corresponds to using the same user embedding for each session. Rather, how often the user embedding is updated is a trade-off between recommendation effectiveness and model efficiency. In this work, we consider the problem of learning user embeddings that are updated based on the sequential consumption of past sessions and the current context.

Regarding consumption, prior work has shown that music consumption is highly driven by recency [4], as users tend to repeat the same tracks often [4, 18]. Regarding context, it has been established that the tracks a user listens to are often context-dependent, such as based on the time of the day [17], location [51], weather [67], and current season [63]. However, these studies on contextual dependency have been done on small datasets, and do not investigate its impact on recommendation effectiveness.

Motivated by the above, we ask the following research questions:

RQ3 To what degree does music consumption depend on context?

RQ4 To what extent can sequential and context-dependent user embeddings better anticipate a user's music consumption?

To answer **RQ3**, we explore historical data from an online music streaming service, containing information about the tracks streamed by a sample of 200,000 users over a two month period. In our analysis, we consider two types of contexts: the time of the day (temporal context), and the device used for music streaming (device context), such as mobile, desktop, speaker, etc.

For both sessions associated with an individual user and across all users, we find that the tracks within a session are more similar to the tracks in sessions of the same context, compared to sessions from a different context. Furthermore, we also find that tracks deviating highly from a user's average preferences, represented by what tracks they usually listen to, are more likely to be skipped by the user.

The findings from **RQ3** support the idea of developing sequential and context-aware models for representing user preferences. To this end, we address **RQ4** by introducing a new recurrent model that generates user embeddings matching the user's preference based on their current context (in the current session) and from the sequence of past consumed sessions. Our model is trained to maximize the cosine similarity between the user embedding and tracks played during a given session. We find that a highly effective way to learn the user embedding is by fusing a global context-independent embedding (representing the user's average preferences) with a learned sequential and contextual offset embedding (representing the sequential and contextual deviations to the user's average preferences). We experimentally compare our model to state-of-the-art embedding-based models in a range of ranking tasks, where we observe improvements in ranking effectiveness upwards of 10%. Interestingly, we find that the largest gains occur in the least frequent contexts, highlighting the model's ability to accurately learn the contextual deviations between sessions.

Chapter 4: Modelling Sequential Music Track Skips using a Multi-RNN Approach

In **RQ1-4** we considered the problem of music recommendation within or between sessions, with the aim to recommend music tracks that the user is unlikely to skip. To this end, understanding and inferring the skip behaviour of users within a session is very important, and was in fact a competition challenge for the WSDM Cup 2019 [13]. In this challenge, a session was split in two halves, such that the task was to predict the individual track skips occurring in the last half of the session, based on the skips made by the user in the first half. For all tracks in the session, track features (e.g., popularity or musical features like strength or flatness) were available. The only difference between the two halves was the existence of user feedback in the form of skips. This problem shares similarities with the general problem of sequence-to-sequence prediction [78], but differs from it in that only the user feedback is unknown; whereas, in typical sequence-to-sequence problems, the whole predicted sequence is unknown. Interestingly, it has been observed that the skip behaviour of users is not entirely dependent on the actual track, but also largely depends on whether the user skipped the previous track [13].

Motivated by the above, we raise the following research question:

RQ5 To what extent is future sequential skip behaviour predictable by the past?

To answer **RQ5**, we propose an encoder-decoder model based on two distinct stacked recurrent neural networks (RNNs) using long short-term memory (LSTM) units. This encoder-decoder architecture is a type of neural architecture that is often used for sequence-to-sequence problems, such as machine translation [23, 86].

Our model was the second best performing model in the WSDM Cup 2019 competition (out of 45 teams), with the best performing model [91] also being based on a similar encoder-decoder architecture. We investigated how the model accuracy differs between predicting whether the first track was skipped compared to the average accuracy across all tracks in the second half of the session. We observed that it was notably easier to predict for the first track (0.807 accuracy) compared to the whole second half (0.641 mean average accuracy). This highlights the difficulty of predicting the skip behaviour far out in the future, as if otherwise the accuracy of the two settings should not be drastically different.

1.1.2 Fact-Checking

Chapter 5: MultiFC: A Real-World Multi-Domain Dataset for Evidence-Based Fact Checking of Claims

Automatic fact-checking is the task of predicting the factuality of a claim, typically based on associated evidence and metadata. The evidence is often automatically collected from external knowledge sources, such as search snippets from a search engine [2, 68, 79]. However, existing datasets consist of either a small amount of naturally occurring claims [60, 94] or artificially constructed claims [80].

Motivated by the above, we pose the following research question:

RQ6 How can a large dataset of real-life claims with accompanying evidence be created, to aid in the research of automatic fact-checking?

To answer **RQ6**, we built automatic crawlers for 26 active fact-checking websites.[1] From each website, the crawlers automatically extract the claim, its associated factuality label, and any accompanying metadata that is made available by the individual websites (e.g., tags, speaker name, and publication date). The total crawling resulted in a dataset consisting of 34,918 claims and it was the largest dataset of its kind at the time of publication. To enrich the dataset with evidence, we used the claims verbatim as queries to the Google search API, from which we crawled the top ten retrieved results. To verify the usefulness of the dataset, we trained a state-of-the-art fact-checking model, and ablated the effectiveness impact of using only the claim, including search snippets as evidence, as well as metadata. We found that both evidence and metadata are beneficial for improving the effectiveness of the model.

Chapter 6: Factuality Checking in News Headlines with Eye Tracking

In **RQ6** fact-checking was done using claims, metadata, and associated evidence extracted from the web. In this work, we investigate other modalities of evidence, which can be used to determine the factuality of a claim. Specifically, we consider whether data from eye-tracking can be used to infer the factuality of a claim, as eye-tracking has previously been used in information retrieval to infer relevance [1, 14, 15, 45, 58, 69]. Additionally, eye-tracking has been used to investigate how users engage with news content, where it has been observed that users tend to read false news faster [26], as well as putting more visual attention on credible news posts [77]. These observations establish a relation between a person's reading behaviour and the factuality and credibility of the read material.

Motivated by the above, and focusing on the domain of news, we raise the following research question:

RQ7 To what extent can the factuality of a news headline be inferred using only eye-tracked data?

[1] `https://reporterslab.org/fact-checking/`

To answer **RQ7**, we conducted a user study where the participants were eye-tracked while reading news headlines that are either true or false. The headlines were crawled from a reputable local newspaper and a subset of these were manually falsified using a set of consistent semantic transformations. The participants were eye-tracked while reading the headlines, and five different measures were collected for each headline: the total gaze duration, total fixation duration, total fixation count, average fixation duration, and first fixation duration. Fixation corresponds to a stable eye position within a dispersion threshold, above a duration threshold, and gaze is the cumulative duration of a sequence of consecutive fixations. For inferring the factuality of a headline, we proposed an ensemble model that combines the average factuality prediction of a set of participants to produce the final prediction. The prediction for each participant was modelled as an average of two simple second order logistic models. We chose simple models due to the low amount of available data. We found that the ensemble model over all the participants obtained a mean AUC of 0.69, whereas using only a single participant led to an AUC of 0.55. Thus, it is possible to infer the factuality of a news headline using only eye-tracked data, but the effectiveness is highly dependent on having data from multiple people.

Chapter 7: Neural Check-Worthiness Ranking with Weak Supervision: Finding Sentences for Fact-Checking

Automatic fact-checking methods are trained on claims typically deemed interesting by (reputable) news sources or fact-checking websites, and are as such often manually selected for further fact-checking. The task of check-worthiness prediction is to develop automatic methods for filtering texts, e.g., transcribed political speeches and debates, by assigning a score to each sentence [46]. This score aims to reflect the degree to which a sentence requires fact-checking.

Most existing research, at the time of publication of this article, had focused on using hand-crafted features to predict check-worthiness, such as bag-of-words representations, sentiment scores, and embedding averages [24, 46, 48, 65], rather than representation learning approaches using recurrent neural networks or transformers. Based on the learned check-worthiness scores, a ranked list can be generated for prioritizing which sentences should be fact-checked.

The choice of using models based on hand-crafted features may be due to the limited training data available [61], as more complex models may be more prone to overfitting if used with limited training data. Check-Worthiness is a domain with high availability of data, but small amounts of labelled data, and for such domains weak labelling has been used successfully [21, 62, 89]. Weak labelling is the process of using an existing classifier to get low-quality labels, also known as weak labels, on a typically large amount of currently unlabelled data. The main idea is that these weak labels can then be used as training data, to improve model generalizability.

Motivated by the above, we ask the following research question:

RQ8 Can weak supervision be used for making check-worthiness predictions more accurate?

To answer **RQ8**, we extract a large number of sentences originating from political speeches and debates from the American Presidency Project.[2] These are labelled using ClaimBuster [46], an existing check-worthiness method with a publicly available API.[3] We use a recurrent neural network model that is pretrained on the weakly labelled data, which we evaluate with and without the weakly labelled data. We experimentally show that our model is more effective than state-of-the-art baselines, and that using the weakly labelled data significantly improves effectiveness. While our model greatly outperforms the weak labeller, the only baselines benefiting from the weakly labelled data do not perform better.

Chapter 8: Fact Check-Worthiness Detection with Contrastive Ranking

Existing methods for check-worthiness prediction are trained as a classification task [24, 29, 46, 48, 65, 85], even though they are typically evaluated as a ranking task. As an extension to the model proposed for answering **RQ8** in the previous chapter, we consider how a ranking objective could be incorporated during training, as formulated by the following research question:

RQ9 How can ranking be part of training check-worthiness prediction models?

To answer **RQ9**, we are motivated by the finding in previous work showing a large term overlap between claims and non-claims [56]. Because of this, we posit that check-worthiness models may face difficulties differentiating between highly similar sentences with opposing labels. To this end, for each sentence in our dataset, we find the nearest semantically similar sentences with opposing labels, denoted as contrastive sentences, and we use these as a set of tuples for training. In addition to the standard cross entropy classification loss of our model, we extend it with a hinge ranking loss that better learns to separate the constrastive sentences. We experimentally validate that including the ranking objective on contrastive sentences significantly improves ranking effectiveness, compared to our previous model.

1.1.3 Speed Reading

Chapter 9: Neural Speed Reading with Structural-Jump-LSTM

For the recommendation and fact-checking problems considered so far, we have proposed sequential models for inference and representation learning. Common to these models is that they all consist of recurrent neural networks for processing a sequence of inputs in its entirety. We now consider the problem of whether it is

[2] `https://web.archive.org/web/20170606011755/http://www.presidency.ucsb.edu/`

[3] `https://idir.uta.edu/claimbuster/api/`

necessary to process every input, or whether parts can be ignored without compromising effectiveness. This has been explored in what is called ”speed reading”, which is based on solving text-based tasks using sequential models with the additional goal of making as few state updates as possible in the recurrent model.

Speed reading tasks have traditionally been solved by two types of models. The first type is jump-based models [22, 87, 88], which during reading can choose to jump a certain number of steps ahead in the sequence, or terminate the reading of the sequence entirely when enough information is obtained to solve the task. The second type is skip (or skim) based models [16, 75], which in addition to a full state update, can choose to either ignore the current input, thereby not making any state updates, or to skim the input and make a reduced state update. Common to all models is that they make the decision to ignore part of the sequential input based on the current and previous inputs, but do not utilize any inherent structure in the sequence.

Our motivation is that, in sequences such as text, punctuation is a type of inherent structure that humans use to guide our reading behaviour. Therefore, punctuation could potentially be utilized to determine how the jumps could be done in speed reading text. Inspired by this, we raise the following research question:

RQ10 To what extent can inherent text structure be used for defining the jumps in a speed reading model?

To answer **RQ10**, we propose a new hybrid speed reading model, Structural-Jump-LSTM, which combines both jumping and skipping of an input. The jumping is based on exploiting the punctuation structure, such that a jump is made towards either a comma, the end of a sentence (.!?), or the end of the document. We evaluate our model empirically in text classification and question-answering, and compare it against state-of-the-art speed reading models. We find that our model obtains the overall lowest number of state updates, corresponding to processing the fewest number of sequential inputs. Additionally, we find that speed reading models can often produce more accurate predictions than processing the entire sequence (i.e., full text), due to better generalization, which has similarly been observed in related work [75, 87].

1.2 Summary of Contributions

This thesis makes a number of contributions for sequential problems faced in music recommendation, fact-checking, and speed reading. We summarize the contributions below:

- The first contribution is a study on diversity in sequential recommendation, using two notions of diversity related to popularity and user personalization. To this end, we first propose and evaluate multiple rankers of increasing complexity to study how their complexity impacts the diversity and relevance of the recommendations. Next, to increase the diversity, we investigate different diversification methods to explore their trade-off between increasing diversity and potentially reducing relevance. We find that rankers of high complexity result in more accurate but less diverse recommendations, while diversification methods enable increasing the diversity with little to no reduction in relevance.

- The second contribution is a study on the impact of context on music consumption, where we find that tracks within a listening session are most similar to the tracks from sessions in the same context. Motivated by this, we propose a new sequential model for dynamically generating user embeddings adapting to the contextual deviations from a user's general music preferences. Compared to state-of-the-art embedding-based baselines, we find modelling the contextual deviations to be effective, as seen by ranking improvements of upwards of 10% in a range of ranking tasks.

- The third contribution is an investigation of the extent to which a user's sequential skip behaviour in a listening session is predictable by past skips. To this end, we propose an encoder-decoder model for predicting future unknown skips in a sequence of known recommended tracks. We show that as less recent skip information is available, the accuracy drops significantly, highlighting that the recommended track is not the only factor affecting the act of skipping.

- The fourth contribution is the construction of the largest-to-date fact-checking dataset of naturally occurring claims crawled from 26 active fact-checking websites. The claims are accompanied by evidence pages retrieved from a search engine, using the claims as queries, as well as rich metadata. We experimentally highlight the benefits of utilizing both the evidence and metadata, as seen by their impact on improving effectiveness.

- The fifth contribution is a study of how well factuality of a headline can be determined exclusively using eye-tracking data. The eye-tracking data was obtained from a user study where the participants were eye-tracked while reading factually true and false news headlines. We find that when eye-tracking data was pooled from multiple participants using an ensemble approach, factuality could be reasonably predicted with an AUC of 0.69, highlighting that eye-tracking can be used as a new modality for fact-checking methods.

- The sixth contribution is a new model for detecting check-worthy sentences for fact-checking. We find that training neural models in this domain is heavily limited by small amounts of training data, to which end we propose a strategy for using weak supervision, which significantly improves effectiveness.

- The seventh contribution is an improved model for detecting check-worthy sentences. Motivated by the observation of a large term overlap between claims and non-claims, which are highly similar to check-worthy and non-check-worthy sentences, we propose a model with a ranking-based objective, that better separates sentences with high semantic overlap, but opposing labels.

- The eighth contribution is a new speed reading model, which utilizes the inherent punctuation structure of text for learning how to ignore significant parts of the input sequence, while being equally or more effective than processing the entire sequence.

1.3 Future Work

Based on the contributions presented in this thesis, we outline some potential directions for future work below.

User effort as a measure of quality for sequential recommendation

When evaluating the recommendations of a given recommender system, we normally aim to optimize the relevance of the recommendations. For music recommendation, this means minimizing the number of skips done by a user. This measure of recommendation quality is potentially flawed, because not all skips require the same amount of user effort. We can imagine at least three scenarios:

- The listening device has the screen turned off and is simply used for playing music, in which case turning on the screen to skip a track takes a moderate amount of effort;

- The user is actively using the device but does not have the music player open. In this case skipping a track takes less effort than in the previous example;

- The user has just skipped a track, in which case an immediate subsequent skip would take very little effort.

Based on these scenarios, we posit that collecting basic information about the state of the listening device during a session would allow estimating an approximate effort level of a skip. Rather than training recommender systems to minimize the number of skips, an alternative task would be the minimization of user effort, which could potentially better correlate with user satisfaction in passive listening sessions.

Handling bias in sequential recommendation

When training sequential recommender models we currently assume that relevance feedback is purely dependent on the recommended item, even though this assumption is partly violated by the sequential dependencies between the feedback signals.

One such example of a sequential dependency is the finding that users are much more likely to skip if they have just made a previous skip [13]. If these dependencies can be reliably modelled, they could be included for correction during training, which could lead to more effective recommender systems. This can be seen as a bias correction for sequential models comparable to the debiasing done for list-wise recommendation in unbiased learning to rank [50].

Efficient feedback-based re-ranking for sequential recommendation

Deploying a sequential recommender system that incorporates immediate feedback from a user requires a re-ranking after each interaction. To accommodate this, research into highly efficient sequential recommender systems is required and worth investigating. For non-sequential recommender systems, very efficient hashing-based methods have been investigated, where users and items are represented as hash codes [33, 38], which require very little storage and enable very fast distance computations. However, hashing-methods have so far not been investigated for the domain of sequential recommendation. We posit that this is a direction worthy of further investigation.

Is automatically collected evidence sufficient for determining the factuality of a claim?

In this thesis, in the context of fact checking, we considered the usage of automatically collected evidence as returned by a search engine when using claims as queries. While utilizing this evidence significantly improves the effectiveness of the factuality prediction, compared to only using the claim, many claims still remain difficult to fact check correctly. However, it has not currently been investigated whether this difficulty is due to insufficient evidence, or lack of better modelling, or inherent difficulty of the claim itself. To this end, it would be interesting to perform a user study of how well human assessors are able to determine claim factuality using only the same evidence as used by the fact-checking model. Additionally, it would provide a gold standard of human performance in the setting of evidence-based fact-checking.

1.4 List of Publications

The following published articles are included as chapters of this thesis (* denotes equal contribution):

- Christian Hansen, Rishabh Mehrotra, Casper Hansen, Brian Brost, Lucas Maystre, Mounia Lalmas (2021). Shifting Consumption towards Diverse Content on Music Streaming Platforms. In WSDM, pages 238-246. [44].

- Casper Hansen, Christian Hansen, Lucas Maystre, Rishabh Mehrotra, Brian Brost, Federico Tomasi, Mounia Lalmas (2020). Contextual and Sequential User Embeddings for Large-Scale Music Recommendation. In RecSys, pages 53-62. [31].

- Christian Hansen, Casper Hansen, Jakob Grue Simonsen, Stephen Alstrup, Christina Lioma (2019). Modelling Sequential Music Track Skips Using a Multi-RNN Approach. In WSDM Cup. [42].

- Isabelle Augenstein, Christina Lioma, Dongsheng Wang, Lucas Chaves Lima, Casper Hansen, Christian Hansen, Jakob Grue Simonsen (2019). MultiFC: A Real-World Multi-Domain Dataset for Evidence-Based Fact Checking of Claims. In EMNLP, pages 4685-4697. [7].

- Christian Hansen, Casper Hansen, Jakob Grue Simonsen, Birger Larsen, Stephen Alstrup, Christina Lioma (2020). Factuality Checking in News Headlines with Eye Tracking. In SIGIR, pages 2013-2016. [43].

- Casper Hansen, Christian Hansen, Stephen Alstrup, Jakob Grue Simonsen, Christina Lioma (2019). Neural Check-Worthiness Ranking with Weak Supervision: Finding Sentences for Fact-Checking. In Companion Proceedings of WWW, pages 994-1000. [29].

- Casper Hansen, Christian Hansen, Jakob Grue Simonsen, Christina Lioma (2020). Fact Check-Worthiness Detection with Contrastive Ranking. In CLEF, pages 124-130. [37].

- Christian Hansen, Casper Hansen, Stephen Alstrup, Jakob Grue Simonsen, Christina Lioma (2019). Neural Speed Reading with Structural-Jump-LSTM. In ICLR. [40].

Furthermore, in addition to the research presented in this thesis, articles have been published in the following areas: hashing-based learning for similarity search and recommendation [30, 32, 33, 34, 38], text representation and classification [28, 35, 36, 59, 83], educational datamining [3, 39, 41], and health-oriented modelling [27, 49, 57]. These articles are listed below:

- Christian Hansen*, Casper Hansen*, Jakob Grue Simonsen, Christina Lioma (2021). Projected Hamming Dissimilarity for Bit-Level Importance Coding in Collaborative Filtering. In WWW, in press. [38].

- Christian Hansen*, Casper Hansen*, Jakob Grue Simonsen, Stephen Alstrup, Christina Lioma (2021). Unsupervised Multi-Index Semantic Hashing. In WWW, in press. [30].

- Dongsheng Wang*, Casper Hansen*, Lucas Chaves Lima, Christian Hansen, Maria Maistro, Jakob Grue Simonsen, Christina Lioma (2021). Multi-Head Self-Attention with Role-Guided Masks. In ECIR, in press. [83].

- Espen Jimenez Solem, Tonny Studsgaard Petersen, Casper Hansen, Christian Hansen, et al. (2021). Developing and Validating COVID-19 Adverse Outcome Risk Prediction Models from a Bi-national European Cohort of 5594 Patients. In Scientific Reports 11 (1), pages 1-12. [49].

- Lucas Chaves Lima*, Casper Hansen*, Christian Hansen, Dongsheng Wang, Maria Maistro, Birger Larsen, Jakob Grue Simonsen, Christina Lioma (2021). Denmark's Participation in the Search Engine TREC COVID-19 Challenge: Lessons Learned about Searching for Precise Biomedical Scientific Information on COVID-19. In TREC COVID-19 Challenge. [57].

- Casper Hansen*, Christian Hansen*, Jakob Grue Simonsen, Stephen Alstrup, Christina Lioma (2020). Content-aware Neural Hashing for Cold-start Recommendation. In SIGIR, pages 971-980. [33].

- Casper Hansen*, Christian Hansen*, Jakob Grue Simonsen, Stephen Alstrup, Christina Lioma (2020). Unsupervised Semantic Hashing with Pairwise Reconstruction. In SIGIR, pages 2009-2012. [34].

- Casper Hansen, Christian Hansen, Jakob Grue Simonsen, Christina Lioma (2019). Neural Weakly Supervised Fact Check-Worthiness Detection with Contrastive Sampling-Based Ranking Loss. In CLEF-2019 Fact Checking Lab. [36].

- Casper Hansen, Christian Hansen, Stephen Alstrup, Jakob Grue Simonsen, Christina Lioma (2019). Contextually Propagated Term Weights for Document Representation. In SIGIR, pages 897-900. [28].

- Casper Hansen, Christian Hansen, Jakob Grue Simonsen, Stephen Alstrup, Christina Lioma (2019). Unsupervised Neural Generative Semantic Hashing. In SIGIR, pages 735-744. [32].

- Christian Hansen, Casper Hansen, Stephen Alstrup, Christina Lioma (2019). Modelling End-of-Session Actions in Educational Systems. In EDM, pages 306-311. [39].

- Rastin Matin, Casper Hansen, Christian Hansen, Pia Mølgaard (2019). Predicting Distresses using Deep Learning of Text Segments in Annual Reports. In Expert Systems With Applications (132), pages 199-208. [59].

- Casper Hansen, Christian Hansen, Jakob Grue Simonsen, Christina Lioma (2018). The Copenhagen Team Participation in the Check-Worthiness Task of the Competition of Automatic Identification and Verification of Claims in Political Debates of the CLEF2018 CheckThat! Lab. In CLEF-2018 Fact Checking Lab. [35].

- Casper Hansen, Christian Hansen, Stephen Alstrup, Christina Lioma (2017). Smart City Analytics: Ensemble-Learned Prediction of Citizen Home Care. In CIKM, pages 2095-2098. [27].

- Stephen Alstrup, Casper Hansen, Christian Hansen, Niklas Hjuler, Stephan Lorenzen, Ninh Pham (2017). DABAI: A data driven project for e-Learning in Denmark. In ECEL, pages 18-24. [3].

- Christian Hansen, Casper Hansen, Niklas Hjuler, Stephen Alstrup, Christina Lioma (2017). Sequence Modelling For Analysing Student Interaction with Educational Systems. In EDM, pages 232-237. [41].

Chapter 2

Shifting Consumption towards Diverse Content on Music Streaming Platforms

Christian Hansen, Rishabh Mehrotra, Casper Hansen, Brian Brost, Lucas Maystre, Mounia Lalmas (2021). Shifting Consumption towards Diverse Content on Music Streaming Platforms. In WSDM, pages 238-246. [44].

Shifting Consumption towards Diverse Content on Music Streaming Platforms

Christian Hansen*
University of Copenhagen
chrh@di.ku.dk

Rishabh Mehrotra
Spotify
rishabhm@spotify.com

Casper Hansen*
University of Copenhagen
c.hansen@di.ku.dk

Brian Brost
Spotify
brianbrost@spotify.com

Lucas Maystre
Spotify
lucasm@spotify.com

Mounia Lalmas
Spotify
mounia@acm.org

ABSTRACT

Algorithmic recommendations shape music consumption at scale, and understanding the impact of various algorithmic models on how content is consumed is a central question for music streaming platforms. The ability to shift consumption towards less popular content and towards content different from user's typical historic tastes not only affords the platform ways of handling issues such as filter bubbles and popularity bias, but also contributes to maintaining a healthy and sustainable consumption patterns necessary for overall platform success.

In this work, we view diversity as an enabler for shifting consumption and consider two notions of music diversity, based on taste similarity and popularity, and investigate how four different recommendation approaches optimized for user satisfaction, fare on diversity metrics. To investigate how the ranker complexity influences diversity, we use two well-known rankers and propose two new models of increased complexity: a feedback aware neural ranker and a reinforcement learning (RL) based ranker. We demonstrate that our models lead to gains in satisfaction, but at the cost of diversity. Such trade-off between model complexity and diversity necessitates the need for explicitly encoding diversity in the modeling process, for which we consider four types of approaches: interleaving based, submodularity based, interpolation, and RL reward modeling based. We find that our reward modeling based RL approach achieves the best trade-off between optimizing the satisfaction metric and surfacing diverse content, thereby enabling consumption shifting at scale. Our findings have implications for the design and deployment of practical approaches for music diversification, which we discuss at length.

KEYWORDS

Recommender systems; Music; Shifting consumption; Diversity

ACM Reference Format:
Christian Hansen, Rishabh Mehrotra, Casper Hansen, Brian Brost, Lucas Maystre, and Mounia Lalmas. 2021. Shifting Consumption towards Diverse Content on Music Streaming Platforms. In *Proceedings of the Fourteenth ACM International Conference on Web Search and Data Mining (WSDM '21), March 8–12, 2021, Virtual Event, Israel.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3437963.3441775

1 INTRODUCTION

Algorithmically generated recommendations power and shape the bulk of music consumption on music streaming platforms. Given the large role of streaming in the music industry, it has become important for music streaming platforms to consider the influence of their recommendations on music consumption in a manner benefiting not only the users, but also artists, and the long term goals of the platform itself. The ability to influence and shift consumption at scale enables system designers to maintain healthy consumption patterns needed for long term platform health and success.

A fundamental characteristic of a music recommendation system that helps platforms shape consumption is its *diversity*. What does diversity means in the context of music recommendation? First, it can facilitate exploration by helping users discover new content or inculcate new tastes [11, 31, 34]. Additionally, it can help the platform spread consumption across artists and facilitate consumption of less popular content. This, in turn, can help counteract rich-get-richer phenomena common throughout the music industry [24]. Finally, it has recently been shown that consumption of diverse music genres is strongly associated with important long-term business metrics, such as user conversion and retention [1].

We formalize our notion of diversity around two central factors that influence music consumption via recommender systems: 1) taste similarity, or how similar a piece of music is to the type of music the user has historically streamed, and 2) popularity, or how many users have recently streamed the piece of content [12]. Based on this, two notions of diversity naturally emerge, one based on the user bias of consumed content, and another based on the global bias of consumed content. From the former point of view, one can achieve diversity and shift consumption by avoiding recommending similar songs to what the user has historically streamed, while in the latter view of diversification, one can shift consumption towards the long tail of consumed music. Focusing on these two notions of diversity enables us to effectively and efficiently drive diversity and influence music consumption, both

*This work was done as part of an internship at Spotify.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

WSDM '21, March 8–12, 2021, Virtual Event, Israel
© 2021 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-1-4503-8297-7/21/03…$15.00
https://doi.org/10.1145/3437963.3441775

at the user- and global- level.

Present Work. We focus on the case of sequential recommendations, and consider four different types of sequential recommenders, or *rankers*, of increasing complexity. We leverage two widely used types of rankers: similarity based and feed-forward neural rankers, and propose two additional rankers, a feedback aware neural attention ranker, and a reinforcement learning (RL) based ranker. This provides us with a wide spectrum of approaches, from simple similarity based rankers to sophisticated reward based RL ranker, and enables us to understand the interplay between model complexity and performance. In support of recent findings that highlight the drop in diversity metrics for models optimized for user satisfaction [15], we investigate how these rankers perform in terms of diversity. Further, we consider four different ways of incorporating diversity in recommendation models: (i) linear interpolation, (ii) submodular diversification, (iii) interleaving based and (iv) reward modeling based on reinforcement learning (RL) ranker.

Overall, our work considers three key questions around (i) the role of two classes of diversity for shifting consumption, and their interplay with user satisfaction, (ii) how four rankers of increasing complexity fare in terms of satisfaction and diversity objectives; and (iii) how four different techniques of incorporating diversity manage the trade-off against user satisfaction.

Overview of results. Looking at music consumption data, we find evidence that users can often be satisfied with recommendations that depart from their historic taste profiles and that are less popular. This underpins the scope for shifting consumption towards diverse content without dissatisfying users. Comparing different rankers, we find strong evidence suggesting that satisfaction centric rankers are heavily biased towards popular content, and content closely resembling user's historic listening activity, which should not be surprising. Interestingly, this bias increases with model complexity, with advanced rankers suffering from this bias to a greater extent.

Among the different diversification techniques, we see that the reward modeling approach for RL model obtains the best trade-off by obtaining a high satisfaction metric and succeeding in surfacing less popular content. For diversity with respect to a user's listening history, we observe that the RL approach performs comparably to the interpolation strategy, with the interpolation strategy offering a wider range of trade-off and subsequently more control over consumption. More interestingly, comparing these results with the ranker comparison on only satisfaction, we observe bigger differences in satisfaction metrics when rankers consider diversity, than when they are only focused on satisfaction.

Taken together, our work sheds light on a central tension between optimizing recommendation models for satisfaction centric objectives versus diversity goals. Developing better rankers results in increasing short term user satisfaction, albeit slightly. However, such models tend to serve less diverse recommendations.

2 RELATED WORK

Retrieving diverse documents has long been recognized as an important challenge in information retrieval [5–7] and for recommender systems [10]. The central problem is that in many applications, it is not sufficient simply to return relevant items, instead the system must account for multiple user intents and needs, in addition to possible redundancy in the content of the returned items. The term *diversity* was first used within information retrieval in [6]. Here a list was considered diverse if it contained items with low similarity to each other. The ranked list was built greedily, with the score of each item being an interpolation of the expected relevance to the user, and the dissimilarity of the item to all previously recommended items in the list. The problem of diversity in list recommendation has in later years received great interest in developing more advanced methods to ensure list diversity [2, 3, 22, 29]. A detailed survey of a variety of methods can be found in [10].

Closely related to diversity is the notion of fairness in recommendations, e.g. [4, 25]. Here we consider diversity from the point of view of the recommended items, e.g. in group fairness, where if the items can be considered to be part of a group, all groups must on average be represented in the final recommendation. This can be extended to marketplace settings, where multiple different stakeholders have requirements for the fairness of the recommendation [15]. Thus, whereas diversity is often considered to be a user centric concern, fairness is item centric, as a fair ranking needs to give equal opportunity for the recommended items.

Whereas existing work on diversity and fairness tends to focus on the ranked list setting, we consider the problem of sequential recommendations of single items. This is a substantially different problem setting, since the user is forced to consume each recommended item, and items introduced to satisfy diversity objectives cannot be as easily ignored by the user if they turn out irrelevant, as in the case of a ranked list.

The interplay between recommender systems and diversity has been popularized in [20], raising public awareness on the so-called "filter bubble" phenomenon. There has been a number of works looking at the effects of recommender systems on the diversity of consumption. A study of a movie recommender system used on a popular e-commerce web site found that the recommendations led to a decrease in sales diversity [8]. By contrast, a study on the effect of recommendations on the YouTube video platform was shown to lead to more diverse consumption [33]. Finally, in the context of music, a strong relationship between consumption diversity and long term platform metrics such as retention and conversion was shown [1]. These finding support the need for properly addressing diversity as part of the recommender system design.

Acting on diversity entails a *formal definition* of diversity. For example, in [1, 28] a setting where items are embedded in a Euclidean space is considered, and diversity is then defined as a function of pairwise distances in that space. Other definitions have been used in [7, 15, 23]. In this work, we consider two operationalizations of diversity, with a focus on simple and practical definitions that are easy to implement in real-world systems.

3 DIVERSITY FOR CONSUMPTION SHIFTING

Our goal is to understand how algorithmic recommendations can help shift consumption through diversity in music consumption. Given the sequential nature of music consumption wherein the user sequentially decides to stream or skip the recommended music, it is not straightforward to recommend a track solely for the purpose of increasing diversity, especially if the track has a low chance of being listened to. Given this complex interplay between relevance

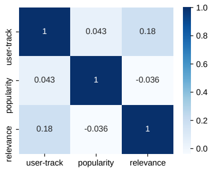


Figure 1: Track level correlation.

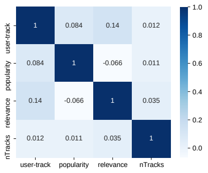


Figure 2: Session level correlation.

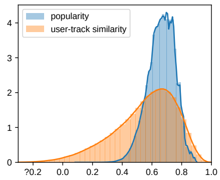


Figure 3: Diversity distributions across tracks.

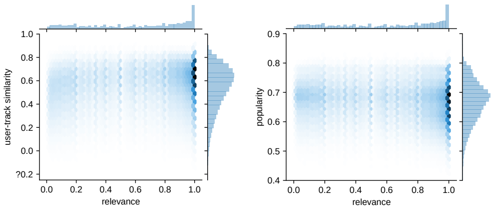


Figure 4: Density of popularity/user-track similarity in relation to relevance across session.

of music to the user, its popularity and the resulting success of diversification, it becomes important to carefully understand the relationship between such concepts. In this work we consider a track to be irrelevant if the user skipped it and otherwise relevant if the user listened to the track. We begin by looking at diversity through the lens of user-track similarity and popularity, and investigate how often are users satisfied with content that either departs from their historic listening habits, or is less popular. Understanding this enables us to underpin the scope of consumption shifting via diversification.

3.1 Quantifying diversity

While numerous ways of defining and quantifying diversity exist, in this work, we are interested in two notions of diversity: 1) diversity *on a global level*, where the diversity is defined as a property of a track t itself ($d(t)$); 2) diversity of a *track as depending on the user* u to which it is recommended ($d(t,u)$), such that the diversity would differ if the same track is recommended to two different users. $d(t)$ encompass a broad set of different notions of diversity, e.g., a high diversity score could be associated with new tracks on the platform, or for tracks of genres rarely listened to by the general user base. Similarly, $d(t,u)$ encompass different notions where the diversity is depending on the user, e.g., a high diversity score could be associated with tracks of artists rarely listened to by a user, or tracks from time periods less familiar to the user.

While the work in this paper can be used for different notions of diversity of the form $d(t)$ or $d(t,u)$, we choose to work on two specific notions of diversity of great importance for music recommendation. For $d(t)$, we consider the global popularity of the track, while for $d(t,u)$, we consider how similar the recommended track is to tracks previously encountered by the user. For ease of notation we always denote the diversity as $d(t,u)$, even though the track level diversity is independent of the user, i.e., $d(t,u') = d(t,u)$ for all users u, u'.

The similarity between a track and tracks previously encountered by the user (denoted as the user-track similarity) is computed as the cosine similarity between a user embedding and a track embedding (see Section 4.1), where the user embedding encodes information from all tracks the user has streamed in the past. The popularity of a track is determined by usage statistics on the platform.

3.2 Analysis of diversity and user satisfaction

We investigate how the notions of diversity, as defined in this work, are related to relevance, and overall engagement measured by session length (measured as the number of tracks within a session). We conduct our analysis on both track- and session- level, and consider a track to be relevant if the user did not skip it. For the track level analysis we use a dataset of 2 million randomly sampled recommended tracks, containing the popularity of the track, the user-track similarity, and the relevance. For the session level analysis we randomly sampled 1 million user sessions, where each session has at least 5 tracks to filter out short sessions. For each session, we log the number of tracks, average popularity, and average user-track similarity across the session, as well as the number of tracks relevant to the user.

Track level: The distribution of popularity and user-track similarity can be seen in Figure 3, and the correlation between the notions of diversity and relevance can be seen in Figure 1. The distribution plots show that users engage with tracks of varying popularity and user-track similarity, but with a large tendency to engage with tracks of both high popularity and user-track similarity. For both distribution plots, the density drops rapidly as popularity/user-track similarity decreases. The correlation plots between the notions of diversity and relevance show that user-track similarity is positively correlated with relevance, which indicates that reducing user-track similarity potentially can harm the user experience. In contrast, the popularity of the tracks is not correlated with relevance, and could likely be reduced without harming the user experience.

Session level: Figure 2 shows the correlation between session popularity, user-track similarity, average relevance of recommended tracks, and number of tracks in the session (session length). We observe that the average popularity is not correlated to either the session length or the average relevance. As seen in the track level analysis, user-track similarity is correlated with relevance, but interestingly it is not correlated with the session length. Figure 4 shows the distribution of both notions of diversity with regards to the average relevance of the session. The highest density is at high popularity/user-track similarity and at fully relevant sessions, but there is considerable density outside this area. Indeed, sessions exist where users are not satisfied with the most popular tracks (upper left side), and there are sessions where they are satisfied with low popularity tracks (lower right side). The same can be observed for user-track similarity.

This analysis motivates that it is possible to shift consumption towards more diverse recommendations without harming user satisfaction, and the typical focus on high popularity/user-track similarity is detrimental for some sessions.

4 RANKERS & DIVERSITY METHODS

We consider the problem of sequential recommendation in a session, where a user consumes a series of recommended music tracks. In this setting, users can either skip or listen to a track. We consider

Table 1: Description of user, track, and user-track combination features used in the neural rankers.

Feature Type	Feature	Description
User	embedding	40 dimensional learnt word2vec vector of user
	country	country of registration for user
Track	embedding	40 dimensional learnt word2vec vector of track
	popularity	normalized popularity of the track
	genres	genres relevant to the track
	acoustic	16 derived acoustic features
	track length	track duration in seconds
User-Track	similarity	cosine similarity between user and track embeddings
	distance	Euclidean distance between user and track embeddings
	genre affinity	affinity for highest overlapping genre between user & track
Playlist	playlist ID	a unique playlist identifier used for learning embeddings

a skipped track as irrelevant, and a listened track as relevant. A session starts with a user selecting a playlist, which consists of tracks with some thematic overlap (e.g., Jazz songs), and is recommended a series of tracks from the playlist, until the user chooses to end the session. We consider two different recommendation scenarios. In the first one, we aim to recommend the tracks a user is most likely to enjoy, and consider four different *rankers* of increasing complexity for this purpose. In the second one, we aim to recommend tracks the user is likely to enjoy, but with the secondary objective that the tracks should also be diverse, where the definition of diversity is detailed in Section 3. To include *diversity* in the recommendation, we explore four different methods to optimize the trade-off between making both relevant and diverse recommendations.

4.1 Preliminaries

We describe the features available to the rankers. This is important as part of the difference between the rankers is to what extent they can make use of the feature space. An overview of the features can be found in Table 1.

Ȩach track is represented as a concatenation of three distinct feature vectors: a **contextual** vector, an **acoustic** vector, and a **statistic** vector. The **contextual** vector is a 40 dimensional real valued vector, which is trained such that two tracks that occur in the same context, will lie close to each other in the vector space [15]. The **acoustic** vector consists of 16 derived features that reflect different acoustic features of the track, e.g., loudness. Lastly, the **statistics** vector contains information on the track length and popularity of the track on the platform. Each user is represented as a weighted average of the **contextual** vectors of the tracks the user has played in the past as described in [15]. The similarity between a track and a user are computed by taking the cosine similarity between the user vector and the track **contextual** vector, as they reside in the same space.

For each user and track pair, there are a number of derived features capturing their relations. The cosine similarity and Euclidean distance between the user and track is computed and used as a feature. Additionally, each user has an affinity for all genres, which is used as a feature by taking the maximum affinity within the track's genres. Lastly, each playlist is represented with a unique identifier, which is used by some of the ranking models for learning playlist specific embeddings during model training. In the next sections, the features are grouped into either: T, which is the combination of the track and user-track features (track level features); or M, which is the combination of the playlist embedding and user features (session level meta features).

4.2 Rankers

We present four different rankers of increasing complexity. The first is based on the cosine similarity between user and track, while the remaining three are learned neural models. An overview of the latter three is provided in Figure 5.

4.2.1 Cosine ranker. This ranker uses the cosine between a track's *contextual* embedding, $e_{track} \in \mathbf{R}^{40}$, and a user's *contextual* embedding $e_{user} \in \mathbf{R}^{40}$: $score_{\text{cosine}} = \frac{e_{track} \cdot e_{user}}{||e_{track}||_2 ||e_{user}||_2}$. A high cosine score indicates that the track is similar to tracks the user has previously consumed on the platform. While being simple, this type of ranker has been used for music recommendation in previous work [1, 9, 15].

4.2.2 Feed forward ranker. This is a neural feed forward network, which takes as input the track level features (T) and session level meta features (M). All the features are concatenated, and the network gives a score for a single track:

$$score_{\text{FF}} = \sigma(W_3 \, \text{ReLU}(W_2 \text{ReLU}(W_1 \, [T \oplus M] + b_1) + b_2) + b_3) \quad (1)$$

where *FF* stands for feed forward, $\oplus$ is vector concatenation, ReLU is the rectified linear unit, and σ is the sigmoid function. The weight matrices (W) and bias vectors (b) have input-suitable sizes and are learned during training. The embedding for the playlist is learned by the network during training. The feed forward network consists of 2 hidden layers with relu activation functions, and a prediction layer using a sigmoid activation function. This prediction corresponds to the probability of a user skipping a track, which is optimized using the cross entropy loss. This model is relatively simple, and computationally efficient. We include it to show how well the score can be computed without considering the user's history directly. Note that the network is indirectly aware of the user's history through the user embedding and the user-track features.

4.2.3 Feedback aware ranker. This ranker is our proposed extension of the feed forward ranker, and incorporates the user's previous sessions to compute a dynamic user embedding. While the two previous models gave a score based on a single track, this model needs to be provided the user's history as input. We first cover how the dynamic user embedding is computed, which consists of two parts: 1) summarising a single session, and 2) summarising all sessions to a final dynamic user embedding.

Summarising a single session. Each session, s, consists of session level meta features, M, and a sequence of tracks (T,R) $\in s$, where T is the track level features and R is a indicator whether the user found the track relevant. The session is summarised using a long short-term memory (LSTM) followed by an attention softmax layer:

$$o_i, h_i = \text{LSTM}(s_i | o_{i-1}, h_{i-1}), \quad \hat{o}_i = \text{ReLU}(W_4 \, o_i + b_4) \qquad (2)$$

$$S = \sum_{i=1}^{|s|} \hat{o}_i \frac{e^{W_5 \, \hat{o}_i + b_5}}{\sum_{j=0}^{|s|} e^{W_5 \, \hat{o}_j + b_5}} \qquad (3)$$

where LSTM denotes an LSTM cell which update the hidden state h and output o. The LSTM cell is initialised by a linear projection of the session meta information such that the session representation can be user and playlist dependent.

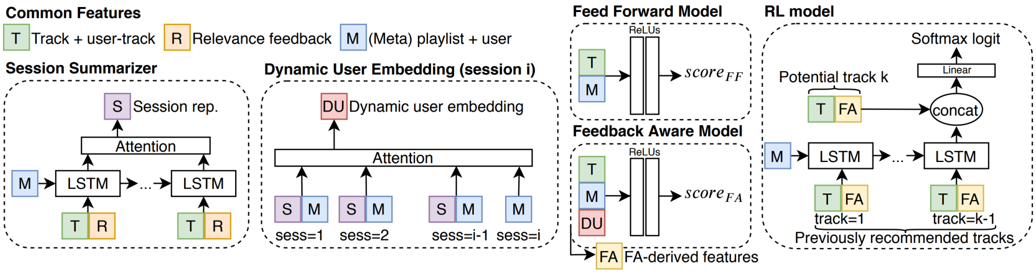


Figure 5: Overview of the neural rankers seen from the perspective of a single track.

Dynamic user embedding. At timepoint i, the users have a set of previous session embeddings, $S_j \in \mathcal{S},\ j \in [1, i-1]$, each having associated meta information M_j. The dynamic user embedding is a summary of all previous session embeddings, conditioned on the current sessions Meta information, M_i. The summarisation of previous sessions is done by attention weighting, where the weighting is based on an interaction vector between the current session meta information, and the historic sessions meta information. The interaction vector [17] is the concatenation, subtraction, and multiplication of the past session and current session meta representations, to represent the representational changes between the sessions. The dynamic user embedding, DU, is then given by

$$\text{interact}(M_i, M_j) = [M_i - M_j \oplus M_i \cdot M_j \oplus M_j \oplus M_i] \tag{4}$$

$$DU = \sum_{j=1}^{i-1} S_j \frac{e^{W_6\,\text{interact}(M_i,M_j)+b_6}}{\sum_{k=1}^{i-1} e^{W_6\,\text{interact}(M_i,M_k)+b_6}} \tag{5}$$

The feedback aware track score is then computed similarly to the feed forward ranker, with the dynamic user embedding (DU) as an additional input:

$$score_{\text{FA}} = \sigma(W_9\,\text{ReLU}(W_8\,\text{ReLU}(W_7\,[T \oplus M_i \oplus DU] + b_7) + b_8) + b_9) \tag{6}$$

where FA stands for feedback aware. Similar to the feed forward ranker, this model is also optimized using the cross entropy loss. This model is still relatively computationally efficient, assuming the dynamic user embedding is pre-computed, which is possible assuming we know the playlists a user is most likely to listen to.

4.2.4 Reinforcement learning ranker. This ranker (RL) is our proposed sampling-based ranker that samples a single track from a set of tracks as the recommendation, which depends on the previous recommended tracks. This process is repeated on the remaining set of possible tracks to produce a ranked list. We formulate the problem of ranking as a standard reinforcement learning problem. We want to find a policy $\pi(t|s)$ that gives the probability of sampling track t given state s. The policy π is learned so it maximises some notion of reward $R(t,s)$, which gives some reward for recommending track t at state s. We therefore have to define the sampling probability $\pi(t|s)$ and the reward $R(t,s)$.

Sampler. Before we cover how $\pi(t|s)$ is computed, we first define how t and s are represented for the RL ranker. t is the track level features (denoted as T previously), but also concatenated with derived features from the feedback aware ranker as explained next. The derived features are the second and last layer of the feedback aware ranker for each track, and we denote this set of derived features as FA. These features are included to provide a richer representation to the RL model, which incorporates the user's past feedback. The state s is a sequence of tracks the user previously has been recommended in the session, in addition to the session meta representation (M). The state is encoded using a stacked LSTM with 2 layers, and initialised based on a linear projection of the session meta information:

$$o_i, h_i = LSTM_{stacked}(s_i|o_{i-1}, h_{i-1}), \quad s_{enc} = o_{|s|} \tag{7}$$

where LSTM$_{stacked}$ is a stacked LSTM with 2 layers, and s_{enc} is the last output of the stacked LSTM. The logit for each track t in the set of possible tracks, $\mathcal{T}$, is then computed as:

$$\begin{aligned} \text{logit}_t = {} & W_{13}\,\text{ReLU}(W_{12}\,[(W_{10}\,s_{enc} + b_{10}) \\ & \oplus (W_{11}\,[T \oplus FA] + b_{11})] + b_{12}) + b_{13} \end{aligned} \tag{8}$$

where both session encoding and track representation are passed through a linear feed forward layer, then concatenated and run through a feed forward layer using a relu activation function, followed by a linear output that gives the unnormalised logit for the track. The unnormalised logit is computed for all tracks in the set of possible tracks, and the sample probability is found by applying a softmax: $\pi(t|s) = \frac{e^{\text{logit}_t}}{\sum_{t' \in \mathcal{T}} e^{\text{logit}_{t'}}}$.

Reward. The reward associated with a sampled track, $t \sim \pi(\cdot|s)$ is defined based on whether the user found the track relevant:

$$R(t,s) = r(t,u) - c \tag{9}$$

where r is a binary relevance function, which is 0 if the user skipped the track and otherwise 1. c is a small constant that ensures that a negative reward is assigned to non relevant tracks. For all experiments c was fixed at 0.1. The model is trained using the REINFORCE algorithm [30].

4.3 Methods for diversity

We describe four methods used to obtain diversity in the recommended tracks. We assume the diversity score can be computed as a function between the track and user, $d(u,t)$, as detailed in Section 3.

4.3.1 Linear interpolation. Given the diversity function $d(u,t)$ and score function $s(u,t)$, the linear interpolation is defined as an α weighted combination of score and diversity:

$$s(u,t)_{diversify} = s(u,t) + \alpha d(u,t) \tag{10}$$

4.3.2 Submodular. Diversity can be introduced by formulating the diversity problem as a submodular set function. Submodular set functions must uphold the following condition:

$$f(X \cup x) - f(X) \geq f(Y \cup x) - f(Y), \quad X \in Y \tag{11}$$

where X and Y are set of items, x is a single item, and f is a real valued function that takes as argument a set. This condition states that a submodular function should have some diminishing return when adding new items to the set. Submodular functions have been used extensively to provide diversity in recommendations [18, 26, 27], as they fit naturally when the set of recommended items should be diverse in regards to some similarity metric between the items. Our notion of diversity (see Section 3) is not naturally submodular, as diversity is a property of either the track itself or the user-track interaction, and thus do not have diminishing returns. To make our notion of diversity submodular, we change the task to recommend tracks of varying diversity. Given a set of recommended tracks τ for user u, we define f:

$$f(\tau, u) = \sum_{t \in \tau} s(u,t) + \frac{\alpha}{|\tau|} \sum_{t' \in \tau \setminus t} \text{abs}(d(u,t) - d(u,t')) \tag{12}$$

where $|\tau|$ is the size of the set τ, and abs is the absolute value. In this setting, we want to recommend the tracks with the highest relevance scores for a given user and that have as different diversity scores as possible, as this maximises the distance between these. This is a NP-hard problem, but can be solved greedily obtaining a near optimal solution [19].

4.3.3 Interleaving. Diversity can be introduced by alternatively recommending tracks with high diversity and high relevance scores. To do this we sort the tracks into two lists, l_{score} and $l_{diversity}$, and sample with probability $1 - \alpha$ from the score list and otherwise from the diversity list at each time step, where α controls the trade-off between relevance and diversity. After each recommendation, the recommended track is removed from both lists.

4.3.4 Reinforcement learning. RL allows us to optimize multiple objectives directly by modifying the reward function. Thus, for the RL ranker we introduce diversity by including a diversity term in the reward function:

$$R(t,s) = r(t,u) - c + \alpha d(t,u) r(t,u) \tag{13}$$

where α is a trade-off parameter between diversity and relevance. Diversity is multiplied with relevance, such that it is only beneficial to recommend diverse tracks when they are relevant to the user.

5 EXPERIMENTAL EVALUATION

We observed strong associations between diversity, relevance and extent of user satisfaction based on the analysis presented in Section 3. The natural follow up question is how the different rankers and diversity methods presented in Section 4 fare, in terms of key satisfaction and diversity metrics, which we investigate next.

5.1 Dataset, metrics and evaluation

We use a dataset from Spotify, a large online music streaming service. The dataset consists of the listening history over a 2 month period of a sample of 1 million of users across 20 million sessions. All users in our sample dataset have at least 5 listening sessions, whereas all sessions have at least 5 tracks. We split users randomly into a training, validation, and testing set (85%, 7.5%, and 7.5%).

Table 2: Performance of rankers relative to the cosine ranker while only optimizing relevance. To avoid revealing sensitive metrics, we introduce a multiplicative factor to the base metrics (Hitrate, NDCG and User-track similarity) reported.

Ranker	Hitrate	NDCG	Popularity	User-track similarity
Cosine	56.006	0.632	1.741	0.584
Feed forward	+2.037%	+2.057%	+4.365%	-10.959%
Feedback aware	+2.553%	+2.848%	+4.078%	-9.247%
RL	+2.703%	+3.165%	+4.538%	-8.048%

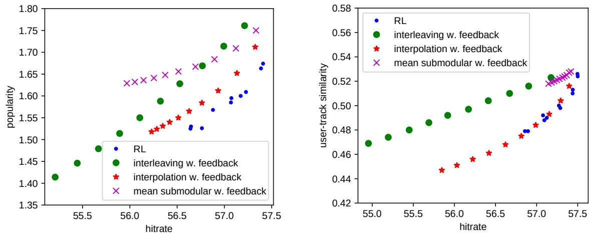


Figure 6: Popularity (left) and Average user-track similarity (right) vs hitrate using the feedback aware ranker.

We measure user satisfaction with the served recommendations using *Hitrate* – the percentage of recommendations relevant to the user (recommendations that the user fully listens to without skipping), as well as Normalised Discounted Cumulative Gain (*NDCG*). For diversity centric experiments, we use as metrics the average popularity of the recommended content (*Popularity*) and average user-track similarity for recommended tracks (*User-track similarity*). To avoid revealing sensitive metrics, we introduce a multiplicative factor to the base metrics (Hitrate, NDCG and User-track similarity) reported.

To keep users engaged in the session from the start, it is important to provide highly relevant initial recommendations. Therefore, given the sequential nature of our problem, we employ a *seed* song based approach, wherein the first track is selected based on relevance, and the diversity metrics are computed on the subsequent recommended tracks. Higher values of *Hitrate* and *NDCG* indicate greater satisfaction, while lower values of *Popularity* and *User-track similarity* indicate more diversity in the served recommendations. We evaluate the rankers on their top 10 recommendations. To have a large and potentially diverse pool of tracks to recommend, we base the evaluation only on sessions with at least 25 tracks.

5.2 Training details

The neural rankers are tuned by choosing the batch sizes within $\{128, 256, 512\}$, and learning rate from $\{0.001, 0.0005, 0.0001\}$. We kept all hidden layers fixed to 50 neurons, and used LSTM sizes of 50 as well. For the feed forward and feedback aware rankers, a batch size of 256 and learning rate of 0.0005 was optimal. For the RL ranker, a batch size of 512 and learning rate of 0.0001 was optimal. To train the RL ranker, as we only have access to logged data, which does not have any propensity scores to allow for off-policy techniques, we use the logged data as a simulator similar to [13, 32]. In our simulation setup, the pool of available tracks is limited to what was originally recommended to the user in a session, and that the user's relevance feedback is the same no matter the order the RL ranker presents the tracks in.

Table 3: **Change (Δ) in hitrate, popularity and user-track similarity in comparison to reinforcement learning optimized only for relevance Table 2.**

	Optimizing popularity		Optimizing user-track sim.	
Method (α)	**Δhit**	**Δpopularity**	**Δhit**	**Δuser-track sim.**
RL (0.1)	-0.212%	-8.324%	-0.037%	-2.235%
RL (0.3)	-0.780%	-12.637%	-0.418%	-7.076%
RL (0.5)	-1.523%	-16.071%	-1.119%	-10.801%
Interpolation (0.1)	-0.675%	-9.231%	-0.395%	-6.145%
Interpolation (0.3)	-1.751%	-14.835%	-1.563%	-12.849%
Interpolation (0.5)	-2.243%	-16.593%	-2.909%	-16.760%
Submodular (0.1)	-0.694%	-6.099%	-0.217%	-1.862%
Submodular (0.3)	-1.992%	-9.451%	-0.421%	-2.793%
Submodular (0.5)	-2.698%	-10.495%	-0.652%	-3.538%
Interleaving (0.1)	-0.916%	-5.824%	-1.073%	-3.911%
Interleaving (0.3)	-2.460%	-14.835%	-2.785%	-8.380%
Interleaving (0.5)	-4.016%	-22.308%	-4.461%	-12.663%

5.3 Comparison of ranking approaches

We begin by investigating the trade-off between model complexity and performance, and investigate how the different rankers fare on diversity metrics when not optimized explicitly for diversity. Table 2 shows the performance of the four rankers on satisfaction and diversity metrics. We observe that the hitrate and NDCG for the rankers follows their computational complexity. The proposed RL ranker have the highest user satisfaction, although we observe a relative small difference in hitrate and NDCG for all neural rankers.

As the increasingly complex rankers lead to higher user satisfaction, they also result in recommendations with a higher average popularity. Most notably, the largest popularity increase occurs when going from the cosine ranker to any of the neural rankers, whereas the popularity difference between the neural rankers is negligible in comparison. For the user-track similarity diversity metric, the cosine ranker will by definition have the largest user-track similarity. However, among the three neural rankers, we observe that the more complex models lead to recommendations that are more similar to what the user has previously encountered. These results suggest that while increased model complexity gives better user consumption predictability, it comes at a cost of decreased diversity. As the hitrate and NDCG both show the same trends, we will focus on only the hitrate for the remaining results.

Note: While seemingly small, a 2-3% gain in offline metrics (e.g. NDCG) has resulted in over 10-15% gain in important online measures of user satisfaction in past A/B tests. This is further supported by prior research that suggests that small changes in NDCG might result in significant changes in online user behevaior [21].

5.4 Comparison of diversity methods

To evaluate the four diversity methods, we compare their performance for introducing diversity against each other, keeping the ranker fixed. For the three methods requiring a track relevance score (interpolation, submodular, and interleaving) we use the feedback aware ranker as the base ranker. These three methods are compared directly against the RL ranker, which is optimized for both relevance and diversity through its reward definition. As optimizing for both relevance and diversity is a trade-off, the results are presented using scatter plots. For the non-RL methods, the trade-off parameter α was chosen as $\alpha \in \{0.05, ..., 0.5\}$ with increments of 0.05. For the RL ranker, we choose $\alpha \in \{0.1, ..., 0.5\}$ with increments of 0.1, and train each configuration twice to explore the variance.

Figure 6 shows the trade-off between hitrate and the diversity metric for the diversity methods, while Table 3 shows the relative values of the hitrate and diversity metrics in comparison to the RL method not optimized for diversity.

Popularity. We observe that the RL method obtains the best trade-off between high hitrate while reducing the average popularity. Linear interpolation obtains the second best trade-off, and interleaving obtains low average popularity at the cost of large reductions in hitrate. Submodular is unable to obtain any large decrease in the average popularity, as larger α values only leads to marginal drops in average popularity. Overall, these results shows a small benefit of using RL to reduce the average popularity, but at the cost of higher computational complexity and training time compared to the simple linear interpolation.

User-track similarity. We observe that the RL method and linear interpolation obtain very similar trade-offs, but that the linear interpolation cover a wider range of trade-offs than the RL method. Diversity by the submodular method results in the worst trade-offs, as the effective user-track similarity reduction is very limited. Similar to the popularity diversity metric, we observe that the interleaving method perform significantly worse than linear interpolation.

Overall, these findings suggest that leveraging RL reward modeling for diversification gives slightly better performance, but interpolation based methods offer a wider range of trade-offs, which provides more flexibility and control to system designers. For submodularity, we observed limited ability to reduce both the popularity and user-track similarity. As described in Section 4.3.2, neither of the diversities are naturally submodular, hence we formulated a submodular function for recommending a sequence of items with varying diversity (as opposed to simply increasing diversity). However, based on the results in our setting, this submodular formulation is less suited to the problem compared to other traditional approaches like interleaving and interpolation.

5.5 Interplay between ranker and diversity methods

We have compared rankers on satisfaction metric, and investigated the effect of the four diversity methods when the ranker was fixed. A natural question to answer is whether the observed trends in diversity methods generalize across all rankers, or does specific diversity methods work with specific rankers. We next investigate this interplay of rankers and diversity methods. For all experiments we use the same choice of α values as done previously.

Popularity. Figure 7 shows the trade-off between average popularity and hitrate for all combinations of rankers and methods introducing diversity. In all cases, the RL ranker is the same and is used as a reference between the plots. We observe that the difference in hitrate from the rankers carries almost directly over for the interpolation and interleaving, while the difference is smaller between the hitrate for the submodular method. Independently of the ranker, the span of average popularity for each of the three diversity methods is approximately the same, showing that the ranker almost entirely influences hitrate. As the average popularity decreases, we observe that the hitrate differences get comparatively smaller than for larger average popularity values. Independent of

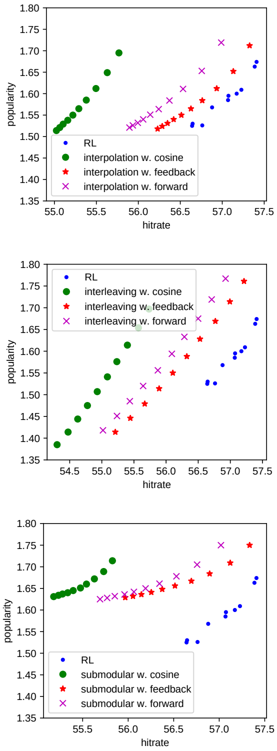


Figure 7: Popularity vs hitrate when varying the ranker across diversity methods.

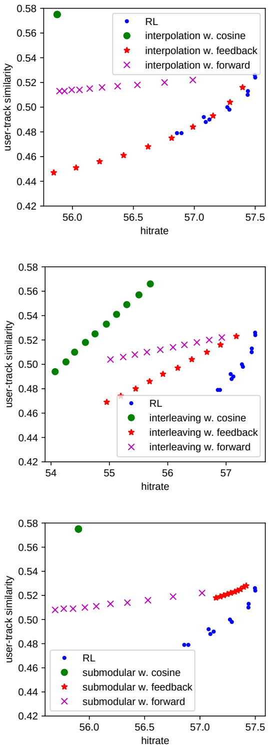


Figure 8: Average user-track similarity vs hitrate when varying the ranker across diversity methods.

the ranker choice, we observe that linear interpolation obtains the best trade-off among the non-RL diversity methods, while RL obtains the best overall trade-off.

User-track similarity. Figure 8 shows the trade-off between average user-track similarity and hitrate for all combinations of rankers and diversity methods. Due to how linear interpolation and submodular both use the diversity metric to subtract from the rank score, they do not work when the diversity metric is the same as the relevance score (as is the case for the cosine), and all values of α therefore leads to the same ranking.

The submodular method again provides the worst trade-offs out of all the diversity methods. When the feed forward ranker is used, the hitrate decrease is notably larger than for the feedback aware ranker, but the effective span of average user-track similarity values is very small for both rankers. For both linear interpolation and interleaving, we observe the difference in hitrate between the feed forward ranker and feedback aware ranker is much greater than the difference observed when only optimizing relevance. While the difference in hitrate between the feed forward and feedback aware ranker is only 0.29 when diversity is not considered (see Table 2), the difference in hitrate can be over 1 depending on the average user-track similarity. This is even though the feedback aware ranker has a slightly higher average user-track similarity when diversity is not considered. Thus, we observe that the choice of ranker can interact with the choice of diversity method non-trivially.

Overall, we observe that RL and linear interpolation work better than interleaving and submodular diversity methods, with both RL and interleaving with feedback aware ranker obtaining approximately the same trade-offs, while the linear interpolation covering a larger span of average user-track similarities. More interestingly, comparing these results with the ranker comparison on only satisfaction (Section 5.3), we observe bigger differences in hitrate when rankers consider diversity, than when they are only focused on satisfaction. This suggests that when one cares only about satisfaction, there exist little difference between the rankers; however when one cares additionally about diversity, the difference between rankers becomes more pronounced.

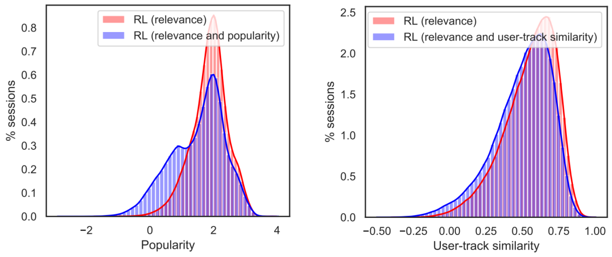


Figure 9: Shift in average session diversity for the RL method ($\alpha = 0.3$) compared to optimizing only relevance

Given the varying complexity of development and deployment of these rankers, this result has big ramifications on the choice of rankers for system designers based on the task at hand.

6 DISCUSSION

Looking at music consumption data, and presented results, we found evidence that not only are users satisfied with relevant recommendations, but also often with recommendations that depart from their historic tastes, or are less popular. Such departure from relevant and popular content allow platforms to broaden the scope of music listening and shift consumption towards the tail and less familiar content. Figure 9 visually depicts this shift in consumption, wherein we observe a significant shift in popularity distribution from unimodal to bimodal when additionally optimizing for popularity diversity, and a slight shift towards lower user-track similarity when optimizing for similarity diversity.

Specifically in the context of music streaming, we posit our findings relates to and builds upon insights on how users consume music. First, recent results suggest that users often have flexible and broad intents when they interact with the music streaming apps [14]. Indeed, the broader the intent, the more we expect the user to be open about music recommendations, which enables the system to shift consumption while still serving satisfying content. Another line of recent research has characterized users as "*specialists*" vs "*generalists*" based on their consumption diversity [1], with generalists preferring diverse sets of music. This highlights the strong preference of some users to prefer diversity, which in turn makes shifting of consumption to less similar or less popular content more amenable. Finally, music streaming applications are essentially multi-stakeholder platforms which connect users and artists [16]. Such platforms need to maintain a healthy balance between user satisfaction and artist exposure goals [15]. A recommender model equipped with consumption shifting ability enables the platform to surface under-served artists, thereby maintaining a healthy balance between consumer and supplier objectives.

On the system design perspective, our findings give system designers practical considerations on the choice of rankers, ways of diversification and serving infrastructure. We argue that the cosine rankers are a good first solution to the recommendation problem – being greedy algorithms, they are quick to deploy, and offer comparable performance to neural and RL rankers if satisfaction is the only goal. However, if diversity is important to consider, and as systems mature and the need for improved models arises, switching to neural ranker makes sense. On the choice between different ways of diversification, the reward modeling based RL method performs better than interpolation for swaying consumption away from popular content, though such methods are non-trivial to productionize at scale. We advocate system designers make this choice based on the underlying infrastructure in place.

Future Work The limitations of this work lead to several next steps. First, while we used logged data to train our RL model, the benefits RL has to offer are more prominent when trained via off-policy training, or a live deployment. Second, there is increasing evidence that propensity for diverse content is an innate user trait, with some users preferring diverse content more than others [1, 15]. This motivates the need for developing user-aware diversification models that personalize the extent to which served recommendations are diverse. Finally, while we explicitly focused on trivial reward combinations, there exist ways to account for richer interactions between objectives. Future work will involve considering richer reward structures to improve performance gains offered by RL approaches.

REFERENCES

[1] Ashton Anderson, Lucas Maystre, Rishabh Mehrotra, Ian Anderson, and Mounia Lalmas. 2020. Algorithmic Effects on the Diversity of Consumption on Spotify. In *The World Wide Web Conference*.

[2] Azin Ashkan, Branislav Kveton, Shlomo Berkovsky, and Zheng Wen. 2015. Optimal greedy diversity for recommendation. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*.

[3] Punam Bedi, Shikha Agarwa, Archana Singhal, Ena Jain, and Gunjan Gupta. 2015. A novel semantic clustering approach for reasonable diversity in news recommendations. In *Computational Intelligence in Data Mining-Volume 1*.

[4] Asia J Biega, Krishna P Gummadi, and Gerhard Weikum. 2018. Equity of attention: Amortizing individual fairness in rankings. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. ACM, 405–414.

[5] Bert Boyce. 1982. Beyond topicality: A two stage view of relevance and the retrieval process. *Information Processing & Management* 18, 3 (1982), 105–109.

[6] Jaime G Carbonell and Jade Goldstein. 1998. The use of MMR, diversity-based reranking for reordering documents and producing summaries.. In *SIGIR*, Vol. 98. 335–336.

[7] Charles L.A. Clarke, Maheedhar Kolla, Gordon V. Cormack, Olga Vechtomova, Azin Ashkan, Stefan Büttcher, and Ian MacKinnon. 2008. Novelty and Diversity in Information Retrieval Evaluation. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '08)*. ACM, New York, NY, USA, 659–666. https://doi.org/10.1145/1390334.1390446

[8] Daniel Fleder and Kartik Hosanagar. 2009. Blockbuster culture's next rise or fall: The impact of recommender systems on sales diversity. *Management science* 55, 5 (2009), 697–712.

[9] Casper Hansen, Christian Hansen, Lucas Maystre, Rishabh Mehrotra, Brian Brost, Federico Tomasi, and Mounia Lalmas. 2020. Contextual and Sequential User Embeddings for Large-Scale Music Recommendation. In *Fourteenth ACM Conference on Recommender Systems*. 53–62.

[10] Matevž Kunaver and Tomaž Požrl. 2017. Diversity in recommender systems–A survey. *Knowledge-Based Systems* 123 (2017), 154–162.

[11] Neal Lathia, Stephen Hailes, Licia Capra, and Xavier Amatriain. 2010. Temporal diversity in recommender systems. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*. 210–217.

[12] Mark Levy and Klaas Bosteels. 2010. Music recommendation and the long tail. In *1st Workshop On Music Recommendation And Discovery (WOMRAD), ACM RecSys, 2010, Barcelona, Spain*. Citeseer.

[13] Elad Liebman, Maytal Saar-Tsechansky, and Peter Stone. 2015. Dj-mc: A reinforcement-learning agent for music playlist recommendation. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, 591–599.

[14] Rishabh Mehrotra, Mounia Lalmas, Doug Kenney, Thomas Lim-Meng, and Golli Hashemian. 2019. Jointly leveraging intent and interaction signals to predict user satisfaction with slate recommendations. In *The World Wide Web Conference*. 1256–1267.

[15] Rishabh Mehrotra, James McInerney, Hugues Bouchard, Mounia Lalmas, and Fernando Diaz. 2018. Towards a fair marketplace: Counterfactual evaluation of the trade-off between relevance, fairness & satisfaction in recommendation systems. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. ACM, 2243–2251.

[16] Rishabh Mehrotra, Niannan Xue, and Mounia Lalmas. 2020. Bandit based Optimization of Multiple Objectives on a Music Streaming Platform. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 3224–3233.

[17] Lili Mou, Rui Men, Ge Li, Yan Xu, Lu Zhang, Rui Yan, and Zhi Jin. 2016. Natural Language Inference by Tree-Based Convolution and Heuristic Matching. In *ACL*. 130–136.

[18] Houssam Nassif, Kemal Oral Cansizlar, Mitchell Goodman, and SVN Vishwanathan. 2018. Diversifying music recommendations. *arXiv preprint arXiv:1810.01482* (2018).

[19] George L Nemhauser, Laurence A Wolsey, and Marshall L Fisher. 1978. An analysis of approximations for maximizing submodular set functions—I. *Mathematical programming* 14, 1 (1978), 265–294.

[20] Eli Pariser. 2011. *The filter bubble: What the Internet is hiding from you*. Penguin UK.

[21] Filip Radlinski and Nick Craswell. 2010. Comparing the sensitivity of information retrieval metrics. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*. 667–674.

[22] Marco Tulio Ribeiro, Nivio Ziviani, Edleno Silva De Moura, Itamar Hata, Anisio Lacerda, and Adriano Veloso. 2015. Multiobjective pareto-efficient approaches for recommender systems. *ACM Transactions on Intelligent Systems and Technology (TIST)* 5, 4 (2015), 53.

[23] Tetsuya Sakai and Zhaohao Zeng. 2019. Which Diversity Evaluation Measures Are "Good"? 595–604. https://doi.org/10.1145/3331184.3331215

[24] Matthew J Salganik, Peter Sheridan Dodds, and Duncan J Watts. 2006. Experimental study of inequality and unpredictability in an artificial cultural market. *science* 311, 5762 (2006), 854–856.

[25] Ashudeep Singh and Thorsten Joachims. 2018. Fairness of exposure in rankings. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 2219–2228.

[26] Choon Hui Teo, Houssam Nassif, Daniel Hill, Sriram Srinivasan, Mitchell Goodman, Vijai Mohan, and SVN Vishwanathan. 2016. Adaptive, personalized diversity for visual discovery. In *Proceedings of the 10th ACM conference on recommender systems*. 35–38.

[27] Sebastian Tschiatschek, Adish Singla, and Andreas Krause. 2017. Selecting sequences of items via submodular maximization. In *Thirty-First AAAI Conference on Artificial Intelligence*.

[28] Isaac Waller and Ashton Anderson. 2019. Generalists and Specialists: Using Community Embeddings to Quantify Activity Diversity in Online Platforms. In *The World Wide Web Conference*. ACM, 1954–1964.

[29] Jacek Wasilewski and Neil Hurley. 2016. Incorporating diversity in a learning to rank recommender system. In *The Twenty-Ninth International Flairs Conference*.

[30] Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning* 8, 3-4 (1992), 229–256.

[31] Yuan Cao Zhang, Diarmuid Ó Séaghdha, Daniele Quercia, and Tamas Jambor. 2012. Auralist: introducing serendipity into music recommendation. In *Proceedings of the fifth ACM international conference on Web search and data mining*. 13–22.

[32] Xiangyu Zhao, Liang Zhang, Zhuoye Ding, Long Xia, Jiliang Tang, and Dawei Yin. 2018. Recommendations with negative feedback via pairwise deep reinforcement learning. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 1040–1048.

[33] Renjie Zhou, Samamon Khemmarat, and Lixin Gao. 2010. The impact of YouTube recommendation system on video views. In *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*. ACM, 404–410.

[34] Cai-Nicolas Ziegler, Sean M McNee, Joseph A Konstan, and Georg Lausen. 2005. Improving recommendation lists through topic diversification. In *Proceedings of the 14th international conference on World Wide Web*. 22–32.

Chapter 3

Contextual and Sequential User Embeddings for Large-Scale Music Recommendation

Casper Hansen, Christian Hansen, Lucas Maystre, Rishabh Mehrotra, Brian Brost, Federico Tomasi, Mounia Lalmas (2020). Contextual and Sequential User Embeddings for Large-Scale Music Recommendation. In RecSys, pages 53-62. [31].

Contextual and Sequential User Embeddings for Large-Scale Music Recommendation

Casper Hansen[*]
University of Copenhagen
c.hansen@di.ku.dk

Christian Hansen[*]
University of Copenhagen
chrh@di.ku.dk

Lucas Maystre
Spotify
lucasm@spotify.com

Rishabh Mehrotra
Spotify
rishabhm@spotify.com

Brian Brost
Spotify
brianbrost@spotify.com

Federico Tomasi
Spotify
federicot@spotify.com

Mounia Lalmas
Spotify
mounia@acm.org

ABSTRACT

Recommender systems play an important role in providing an engaging experience on online music streaming services. However, the musical domain presents distinctive challenges to recommender systems: tracks are short, listened to multiple times, typically consumed in sessions with other tracks, and relevance is highly context-dependent. In this paper, we argue that modeling users' preferences at the beginning of a session is a practical and effective way to address these challenges. Using a dataset from Spotify, a popular music streaming service, we observe that *a*) consumption from the recent past and *b*) session-level contextual variables (such as the time of the day or the type of device used) are indeed predictive of the tracks a user will stream—much more so than static, average preferences. Driven by these findings, we propose CoSeRNN, a neural network architecture that models users' preferences as a sequence of embeddings, one for each session. CoSeRNN predicts, at the beginning of a session, a preference vector, based on past consumption history and current context. This preference vector can then be used in downstream tasks to generate contextually relevant just-in-time recommendations efficiently, by using approximate nearest-neighbour search algorithms. We evaluate CoSeRNN on session and track ranking tasks, and find that it outperforms the current state of the art by upwards of 10% on different ranking metrics. Dissecting the performance of our approach, we find that sequential and contextual information are both crucial.

CCS CONCEPTS

• **Information systems → Recommender systems**; **Music retrieval**.

KEYWORDS

Music Recommendation, User Embeddings, Context, Sequence

ACM Reference Format:
Casper Hansen, Christian Hansen, Lucas Maystre, Rishabh Mehrotra, Brian Brost, Federico Tomasi, and Mounia Lalmas. 2020. Contextual and Sequential User Embeddings for Large-Scale Music Recommendation. In *Fourteenth ACM Conference on Recommender Systems (RecSys '20), September 22–26, 2020, Virtual Event, Brazil.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3383313.3412248

1 INTRODUCTION

Recommender systems are essential for providing an engaging experience and for helping users navigating the vast amounts of content available in online services. Successful recommender systems have to accurately model each user's individual preferences, such that the most relevant content can be presented to the user. In this work, we consider online music streaming services, which have become increasingly popular in the past decade. By letting users access millions of tracks at the click of a button, they are contributing to democratizing access to music. However, in contrast to other well-studied domains (such as recommending books, movies or clothes), music recommender systems face distinctive challenges [30]. Tracks are short, and therefore often consumed together with other tracks; we refer to such a set of tracks listened to in short succession as a session. A given session often contains tracks from the user's recent consumption history [1], suggesting that the *sequence* of sessions captures essential information about users' changing preferences. Additionally, the relevance of tracks is highly contextual, and preferences depend, among others, on the time of the day and the current season [24]. We seek to embrace these distinctive characteristics to produce a better, more accurate model of user preferences. We focus on the following problem: for a given user, we are interested in predicting, at the beginning of a session, which tracks the user will listen to during the session. We assume that we have access to the user's *past consumption* and to information about the *current context.* This formulation of the problem enables generating recommendations that are not only matching the user's global tastes, but are also tailored to the specific context and situation they currently find themselves in. While generic context-aware recommender systems have been studied in the past [28], little work has been focused on music recommendation. We aim to address this gap.

We begin our investigation by exploring a dataset from an online music streaming service, containing detailed information about the tracks streamed during a two months period for a sample of 200,000 users. We define context as the time of the day (morning, afternoon,

[*] This work was done while the first two authors were at Spotify.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

RecSys '20, September 22–26, 2020, Virtual Event, Brazil
© 2020 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-1-4503-7583-2/20/09…$15.00
https://doi.org/10.1145/3383313.3412248

etc.) and the device used to access the service (mobile, desktop, etc.) We find clear evidence that, for a given user, sessions sharing the same context (e.g., sessions happening in the morning) are more similar to each other than to sessions from a different context. We also find that the more the tracks a user listens to during a session deviate from their average preferences, the more likely they are to hit the *skip* button—a negative satisfaction signal. Deviations from the user's average preferences may be due to contextual changes (such as morning vs. evening), but also to preference drifts that are captured in recent sessions. These observations are consistent with our hypothesis: accurately modeling sequential and context-specific intents is important to ensure high user satisfaction across all sessions.

Taken together, these findings support the idea of learning sequence and context-aware models of user preferences. To this end, we introduce *CoSeRNN.*[1] Our starting point is a vector-space embedding of tracks, where two tracks are close in space if they are likely to be listened to successively. Given this space, CoSeRNN models user preferences as a sequence of context-dependent embeddings (points in the track space), one for each session. At its core, it is a variant of a recurrent neural network that takes as input, for each session, the current session context and a representation of the user's past consumption. Given these, the model is trained to output an embedding that maximizes the cosine similarity to the tracks played during the session. Interestingly, we find that the most effective way to produce this embedding is to fuse a long-term, context-independent vector (intuitively capturing a user's average tastes) with a sequence and context-dependent offset (capturing current and context-specific preferences).

We evaluate our approach experimentally against multiple competing baselines on *a*) a session ranking task, where the goal is to discriminate between the current session and previous ones, and *b*) a track ranking task, where the goal is to predict which tracks a user will listen to in the current session. Our approach performs significantly better than competing approaches: we observe gains upwards of 10% on all ranking metrics we consider. We study these results in depth. First, we break them down by context, and observe that CoSeRNN exhibits the biggest gains on infrequent contexts. Second, we perform an ablation study and discover that combining both sequential and contextual information is crucial to achieve high accuracy. In summary, CoSeRNN sucessfully demonstrates the benefits of modeling preferences at the session level.

Setting predictive performance aside, we believe that our design choices also highlight an interesting point in the recommender systems solution space. Broadly-speaking, our method falls within the realm of representation learning, which postulates that low-dimensional embeddings provide an effective way to model users and items [20, 21]. Whereas most of the work in this area has been focused on *jointly* learning user and item embeddings, we choose a different path, and instead take advantage of an existing track embedding space. By decoupling track and user embeddings, and learning the latter based on the former, we ensure interoperability with other models seeking to address problems that are distinct from contextual or sequential recommendations—our focus in this paper. In addition, and similarly to [21], our model does not seek to *directly* predict the individual tracks inside a session; instead, it generates a session-level user embedding, and relies on the assumption that tracks within the session lie inside a small region of the space. Relevant tracks can then be found efficiently using approximate nearest-neighbor search [2]. This choice enables our method to scale to millions of tracks effortlessly.

Outline & Contributions. After briefly discussing related work (Section 2) and describing our dataset (Section 3), we investigate the following two research questions.

RQ1 *Does music consumption depend on context?* By means of simple analyses, we show clear evidence of contextual patterns in music consumption (Section 4) and thus answer the question in the affirmative.

RQ2 *Can sequential and context-dependent user embeddings better anticipate a user's music consumption?* We address this question by presenting CoSeRNN, a sequence and context-aware model of user preferences (Section 5), and demonstrate that it can achieve state-of-the-art results on several prediction tasks (Section 6). We make our implementation of CoSeRNN publicly available.[2]

2 RELATED WORK

Recommender systems can be broadly categorized as using *explicit* or *implicit* feedback, depending on how users are assumed to indicate their preferences [20]. They can be further categorized based on which information is available besides user feedback. This includes content-based [25], sequence-aware [27], context-aware [28], and collaborative filtering recommender systems [20]. In practice, the ideas underlying these various approaches can be combined to match the exact problem setting at hand. Traditionally, matrix and tensor factorization approaches have been widely successful for recommendation tasks [10, 20], but recently deep learning based techniques [37], specifically recurrent neural networks, have received increasing interest, due to their ability to model the sequential nature of user-item interactions effectively [3, 9, 21, 29, 35]. Our model is based on a recurrent neural network, and as such shares similarities with this line of work.

Of particular relevance to us is *session-aware* recommendation [27], where the focus is on modeling users' preferences and intents during a specific session. Early work on session recommendation utilized Markov chains to predict the next action within a session [40], and was later extended to Markov decision processes [31]. However, if higher order models are used, then the state space grows too large and becomes impractical. To this end, recurrent neural network models have proven useful, especially for sequential click prediction tasks [15, 38], where parallel mini-batches and ranking losses lead to large performance increases over earlier work. Other approaches focus on directly exploiting user behaviour to improve performance, e.g., by explicitly modelling repeat consumption [1, 7, 29]. In contrast to previous work, we consider a slightly different setting: we seek to model user preferences at the beginning of a session but *before* observing any user interaction. We also assume access to *explicit* information about the context.

[1] **Co**ntextual and **Se**quential **R**ecurrent **N**eural **N**etwork

[2] Code for CoSeRNN available at https://github.com/spotify-research/cosernn.

The session-based recommender systems described above are similar to the generic next-item recommendation setting [9, 19, 36], but typically session-based methods directly exploit the similarities between items within the same session. Related tasks include predicting the first item in the next session [29], as well as predicting all items in the next session (also known as next-basket recommendation in the e-commerce domain) [34]. However, as the number of possible items grows, predicting every individual item in a session becomes intractable due to the combinatorial number of possibilities. In this paper, we overcome this problem by representing sessions compactly using embeddings. Thus, our setting is similar to the next-item recommendation setting, since ultimately we predict a single embedding representing an entire session.

The idea of predicting the embedding of the next item, rather than the item itself, has been recently investigated [21]. Given an embedding, recommendations are then based on inexpensive similarity computations, which allows for very large item pools [2]. JODIE [21] learns dynamic user and item embeddings through a coupled recurrent neural network, where the item embeddings are based on learning a future user embedding projection. In addition to dynamic embeddings, JODIE also uses static embeddings that represent the long-term stationary properties of users and items, respectively. In contrast to JODIE, our approach does not explicitly project the user embedding, but rather learns it implicitly in the recurrent neural network component of our model. Also, we represent long-term user properties as an explicit combination of all previous sessions, and let the model learn a sequence and context-dependent *offset vector* that is fused with the long-term representation.

2.1 Music Recommendation

Music recommender systems present different challenges compared to recommender systems applied to movies, books, and other products [30]. The major differences are with regards to the duration of an item (e.g., a song is typically much shorter than a movie) and consumption type and intent (music streaming is inherently sequential and highly contextual). Some of these unique characteristics have previously been explored in the setting of playlist generation [4, 8]. Others have investigated the effects of context [6, 14, 33], location [17], and even the weather [26]. However, these studies usually rely on small-scale datasets and do not explore the impact of context on recommendation accuracy and performance. In contrast, our work takes advantage of a large-scale dataset from a leading music streaming service, and evaluates the predictive performance of a context and sequence-aware model on concrete recommendation tasks. Finally, we note that whereas we focus on the sequence of sessions in this work, prior work on the publicly available Spotify Music Streaming Sesssions Dataset [5] addressed the problem of within-session sequencing.

3 DATASET

In this section, we introduce a dataset from Spotify, an online music streaming service. Through Spotify, users have on-demand access to millions of music tracks.[3] We focus on so-called *premium* users, who enjoy an unrestricted, ad-free streaming experience. We consider the listening history of a sample 200,000 users from April 1st to May 31st 2019. We group listening history into sessions, where we define a session as the set of music tracks consumed in a given time interval, such that two sessions are separated by at least 20 minutes of inactivity. On average, users in the dataset have 220 sessions during the two-month period, and each session consists of 10 tracks on average.

Table 1: Summary of the features extracted from a session t contained in the dataset.

Symbol	Description	Domain
D_t	Day of the week	$\{1, \ldots, 7\}$
H_t	Time of the day	$\{0, \ldots, 23\}$
Y_t	Device	$\mathcal{Y}$
N_t	Number of tracks in session	$\mathbb{N}_{>0}$
Δ_t	Time since last session	$\mathbb{R}_{>0}$
z_t	Stream source	$\mathcal{Z}$
$\mathbf{s}_t$	Session embedding, all tracks	$\mathbb{R}^{40}$
$\mathbf{s}_t^+$	Session embedding, *played* only	$\mathbb{R}^{40}$
$\mathbf{s}_t^-$	Session embedding, *skipped* only	$\mathbb{R}^{40}$

3.1 Session-Level Information

Each session is annotated with detailed information, including *a*) the set of tracks played during the session, *b*) which tracks were skipped, *c*) the stream source (user playlist, top charts, etc., collectively denoted $\mathcal{Z}$), *d*) a timestamp representing the start of the session, and *e*) the device used to access the service. We process this information into a set of features, presented in Table 1. The contextual features available at the beginning of the session, D_t, H_t and Y_t, can be categorized into two types:

- **Time context.** We use day of the week D_t and time of the day H_t. Note that even though, in Section 4, we partition sessions by using D_t only, *both* features are used for the model described in Section 5.
- **Device context.** In addition, we consider the device Y_t used by the user to access the service at the beginning of a session. We restrict ourselves to the major devices: $\mathcal{Y}$ = {mobile, desktop, speaker, web, tablet}.

We choose these features as our contextual variables because we believe that they are both important and widely available. Nevertheless, our framework is independent of the particular choice of context and other information (either explicit or implicit) such as mood, activity, or intent can be integrated effortlessly, if available.

The music listened by the user during the session is summarized using three 40-dimensional session embeddings, $\mathbf{s}_t$, $\mathbf{s}_t^+$, $\mathbf{s}_t^-$. These are described in Section 3.3, building upon the description of track embeddings.

3.2 Track Embedding

We embed tracks in a latent semantic space using the *word2vec* continuous bag-of-word model [23] on a set of user-generated playlists. In short, the model learns 40-dimensional real-valued unit-norm embedding for each track, such that two tracks that are likely to co-occur in a playlist are close to each other in the embedding

[3] See: https://newsroom.spotify.com/company-info/.

space, and vice-versa. The similarity between two tracks can then be computed simply by using the cosine similarity between their embeddings. The specific embedding space that we use has previously been shown to work well for music recommendation [22].

It should be noted that, in principle, track embeddings and user embeddings could be learned jointly. By decoupling the two, we simplify the development of multiple models with different goals and improve the scalability of our approach, as discussed in Section 1.

3.3 Session Embedding

The way we represent sessions builds on the track embedding model. In fact, we represent a session simply as an average of the embedding of the tracks it contains. The assumption is that, within a session, tracks cluster around a small region of the embedding space,[4] and as such the average track embedding provides a compact summary of the session's content. We consider three embedding variants for a given session t, all normalized to unit length: s_t represents the average of *all* tracks, while s_t^+ and s_t^- represent the average of *played* and *skipped* tracks, respectively. Considering played and skipped tracks separately provides a more detailed picture of session-level user preferences.

Our definition of session embedding is computationally-efficient: if we also model user preferences by using a unit-norm vector in the same embedding space (as we do in Section 5), we can compute the cosine similarity to a given session's embedding using a dot product. Due to the distributive property of the dot product, the result can be thought of as the average relevance of each track to the user—all using a single dot product.

Finally, we note that, for long sessions, it is no longer clear whether the context stays constant throughout its duration, and whether the tracks played at the end of the session are related to the ones at the beginning. For this reason, we deliberately consider only the first 10 tracks within a session (equal to the average session length), and discard the rest. We are thus effectively understanding, modeling and predicting the *beginning* of a session.

4 EXPLORATORY ANALYSES

In this section, we demontrate the need for contextual models by studying the influence of context on music consumption. We aim to answer the following sub-questions to RQ1, related to context in music consumption:

4.1 How are sessions distributed according to context, and what is the proportion of users experiencing each type of context?
4.2 Does music consumption vary depending on context?
4.3 How similar are sessions across and within different types of contexts?

These questions provide empirical evidence for considering context in music recommendation. Finally, we investigate how user satisfaction relates to how different a session is from a user's average preferences (see Section 4.4)

4.1 Context Distribution Across Sessions

The majority of sessions, as shown in Figure 1, are happening in the afternoon (12pm-5pm) and evening (5pm-8pm). The remaining (46.4%) of sessions are spread out over the remaining time contexts of early morning (6am-9am), morning (9am-12am), late evening (9pm-1am), and night (1am-6am). It is interesting to consider that most users have sessions spanning all types of time contexts, except for the *night* context, which relates to only 76.5% of users. For the device context, we observe that 88.3% of sessions are happening on mobile devices, 8.6% on desktop, and the remaining ones are split across speaker, web, and tablet. However, similar to the time context, a significant amount of users do have at least one session in one or more of the non-mobile devices. These findings highlight that users consume music across multiple contexts, and that even minority contexts are important to consider, since a large number of users do experience those at some point.

4.2 Music Consumption and Context

We investigate if diversity in music consumption depends on context. We collect all tracks appearing in each specific context, and compute the pairwise cosine similarities between tracks within each context. Note that even minority contexts (such as web and tablet) contain millions of plays, meaning that the distributions are well captured for all contexts. Figure 2a illustrates the distributions of pairwise similarities by using boxplots. For the time context, we see that minority contexts, such as early morning and night, have significantly larger variability compared to majority sessions such as afternoon and evening. We observe a similar trend for the device context for all non-mobile contexts compared to mobile. This suggests that users have largely different needs in minority contexts, and as such the user embedding needs to incorporate context in order to reliably estimate user needs.

4.3 Session Similarity and Context

In the previous section, we performed a global analysis across all tracks from every user within a specific context. Now, we analyse if, for a given user, their sessions within the same context are more similar across different contexts. For each user and each session (the *source*), we find the nearest session (the *target*) among all the user's other sessions, and store both the source's and the target's contexts. We then aggregate all the pairs and compute the empirical distribution of the target's context type, conditioned on the source's context type. To correct for the bias induced by the users' distinctive usage patterns (some users might use the service more in some contexts than in others), we subtract from this distribution the marginal probability of each pair of contexts (or, equivalently the empirical distribution obtained when sampling *source* and *target* uniformly at random among the same user's sessions).

Figure 3 displays the result in the form of heatmaps. The positive diagonal tells us that sessions sharing the same context are indeed more similar than sessions sampled at random. Additionally, for time contexts, those occurring close to each other (e.g., night and evening) also often have small positive values, indicating that sessions even from consequent contexts are more similar than random sessions. This analysis highlights that sessions with the same context do share some similarities, which can be exploited to learn better performing contextual user embeddings.

[4] We validate this assumption empirically using track and session ranking tasks in Section 6.

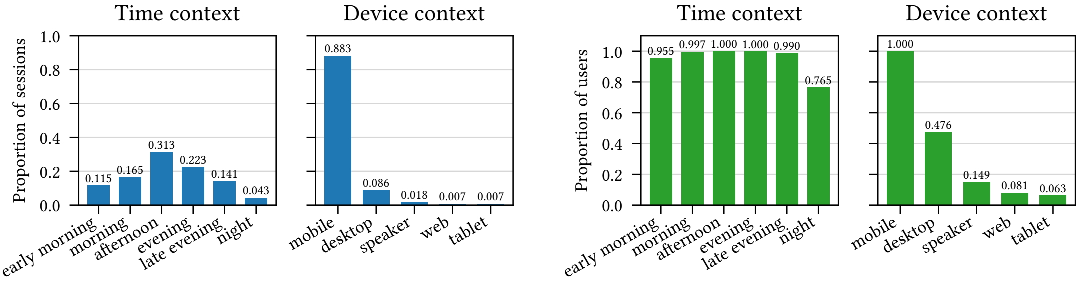
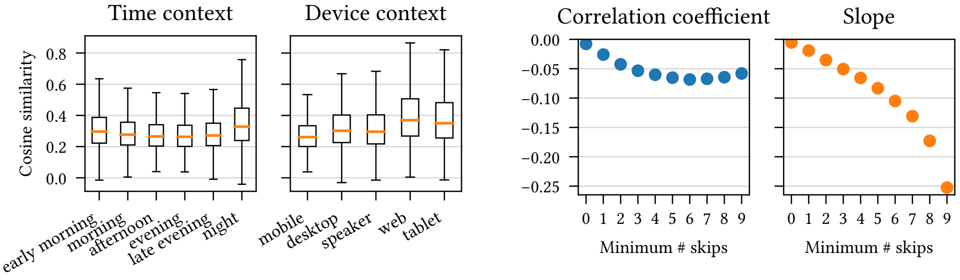


Figure 1: Left: histograms of session context. Right: proportion of users with at least one session within a context.



(a) Homogeneity within contexts.

(b) Cosine similarity and skip rate.

Figure 2: Left: boxplots showing the distribution of pairwise cosine similarity for all tracks occurring within each context. Right: Pearson correlation and regression slope of the relation between skip rate and user-session cosine similarity.

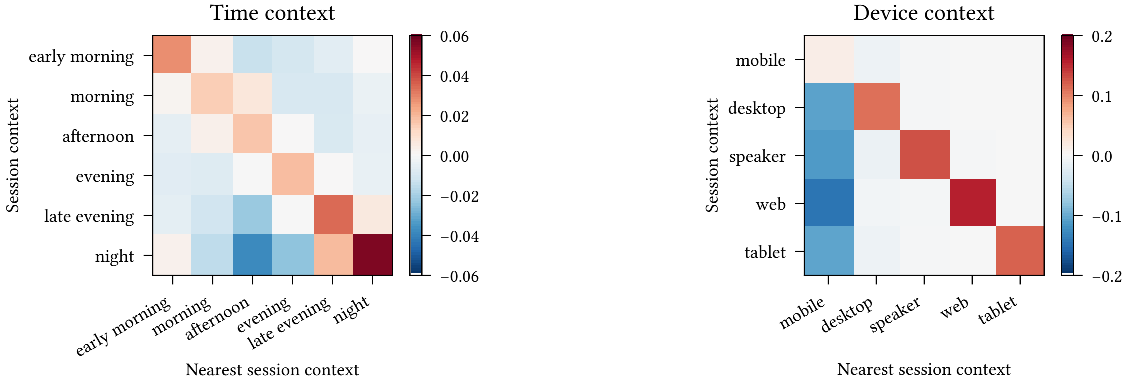


Figure 3: Visualization of the probability difference between sampling sessions ranked by their pairwise cosine similarity or randomly. A positive number corresponds to that pair of context types being more similar than a randomly sampled session.

4.4 Contextual Preferences and Skip Rate

In the previous analyses we found evidence of context being useful for providing a more accurate picture of users' preferences. Now, we consider the influence of a better match between user and session embeddings (i.e., higher cosine similarity) on user satisfaction. As a proxy for satisfaction, we measure the skip rate, i.e., the percentage of skipped tracks within a session. For the purposes of this analysis, we define the user embedding as an average of the embeddings of all their previous sessions. For each session we record the cosine similarity between the user embedding and the current session's embedding, as well as the skip rate of the session.

Figure 2b shows the Pearson correlation coefficient and regression slope between the skip rate and user-session cosine similarity, as a function of the minimum skip rate. By considering sessions with at least k skips (x-axis in the figure), we filter out sessions with low activity, since a very low amount of skips may simply be due to the user not being actively engaged. We observe that both the correlation coefficient and regression slope are negative. This means that, as users skip more often, the user-session similarity decreases. Additionally, both the correlation coefficient and regression slope generally decrease the larger the minimum number of skips is. We expect that, if we were able to anticipate these "unusual" sessions (i.e., sessions that deviate significantly from a users' average preference), we might be able to improve user satisfaction. As we will demonstrate in the next sections, sequence and context-aware models enable us to achieve that goal.

5 CONTEXTUAL AND SEQUENTIAL MODEL

The previous section established the importance of context for understanding users' behaviors. Building on these findings, we now present CoSeRNN, a user-embedding model that captures contextual and sequential preferences at the session level. The aim is to predict, at the very beginning of a session (without observing any explicit action from the user), which tracks will be played during the session, based on features derived from the past consumption history and the current context. Section 5.1 presents the architecture of our model and Section 5.2 discusses the procedure we use to train it.

5.1 Model Architecture

For conciseness, we consider a single user. For a given a session index t, we denote the predicted session-level user embedding as $\boldsymbol{u}_t \in \mathbb{R}^{40}$, and the observed (ground-truth) session embeddings as $\boldsymbol{s}_t^-, \boldsymbol{s}_t^+ \in \mathbb{R}^{40}$, as defined in Section 3.3. The model is trained to maximize the similarity between $\boldsymbol{u}_t$ and $\boldsymbol{s}_t^+$ (this will be made precise in Section 5.2). A diagram of the architecture of our proposed model, CoSeRNN, is provided in Figure 4. At a high level, CoSeRNN models $\boldsymbol{u}_t$ as follows. It uses features about the current context (such as time of the day and device) and features about the last session as input to two RNNs, representing *play* and *skip* behavior. These RNNs combine the input with a latent state, capturing sequential dependencies in the user's consumption habits. Finally, the outputs of the two RNNs are combined and fused with a long-term user embedding.

5.1.1 Notation. We denote a dense (fully-connected) neural network layer by $\mathrm{FC}_g(\boldsymbol{x}) = g(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b})$, where $\boldsymbol{W}$and $\boldsymbol{b}$ are a weight matrix and a bias vector of suitable dimensions, respectively, and g is an activation function. We consider three such functions, *a*) the identity function $\mathrm{id}(\boldsymbol{x}) = \boldsymbol{x}$, *b*) the elementwise rectifier $\mathrm{ReLU}(\boldsymbol{x}) = [\max\{0, x_i\}]$, and *c*) the softmax function $\mathrm{softmax}(\boldsymbol{x}) = [\exp x_i / \sum_j \exp x_j]$. We denote by $f \circ g(x)$ the composition of f and g evaluated at x, i.e., $f(g(x))$, and by $\boldsymbol{x} \oplus \boldsymbol{y}$ the concatenation of the vectors $\boldsymbol{x}$ and $\boldsymbol{y}$.

5.1.2 Input Layers. We start with two feature vectors,

$$\begin{aligned} \boldsymbol{f}_t^+ &= \boldsymbol{c}_t \oplus \boldsymbol{s}_{t-1}^+ \oplus \begin{bmatrix} N_{t-1} & z_{t-1} & \Delta_t \end{bmatrix} \\ \boldsymbol{f}_t^- &= \boldsymbol{c}_t \oplus \boldsymbol{s}_{t-1}^- \oplus \begin{bmatrix} N_{t-1} & z_{t-1} & \Delta_t \end{bmatrix} \end{aligned}$$

where $\boldsymbol{c}_t$ is a concatenation of one-hot encodings of the contextual variables D_t, H_t and Y_t, and all other symbols refer to Table 1. These are input to the *play* and *skip* pathways of the network, respectively. Prior to passing the features to the RNN, we apply a learned nonlinear transformation. This enables the RNN to better focus on modeling latent sequential dynamics. In particular, we apply the following transformation: $\hat{\boldsymbol{f}}_t^+ = \mathrm{FC}_{\mathrm{ReLU}} \circ \mathrm{FC}_{\mathrm{ReLU}}(\boldsymbol{f}_t^+)$, which corresponds to two fully connected layers with ReLU activations. We obtain $\hat{\boldsymbol{f}}_t^-$ by applying the same transformation to $\boldsymbol{f}_t^-$.

5.1.3 Recurrent Layers. Next, we seek to capture and reuse relevant information from the user's history—beyond the last session. We do so by using an RNN with Long Short Term Memory (LSTM) cells [16], and let $(\boldsymbol{o}_t^+, \boldsymbol{h}_t^+) = \mathrm{LSTM}(\hat{\boldsymbol{f}}_t^+ \mid \boldsymbol{o}_{t-1}^+, \boldsymbol{h}_{t-1})$, where $\boldsymbol{o}_t^+$ is the output and $\boldsymbol{h}_t^+$ the hidden state.[5] Similarly, we obtain $(\boldsymbol{o}_t^-, \boldsymbol{h}_t^-)$ from $\hat{\boldsymbol{f}}_t^-$. We learn to combine the outputs $\boldsymbol{o}_t^+$ and $\boldsymbol{o}_t^-$, and then obtain the sequence and context-dependent part of the user embedding, $\hat{\boldsymbol{u}}_t$, as

$$\boldsymbol{o}_t = \mathrm{FC}_{\mathrm{ReLU}} \circ \mathrm{FC}_{\mathrm{ReLU}}(\boldsymbol{o}_t^- \oplus \boldsymbol{o}_t^+), \qquad \hat{\boldsymbol{u}}_t = \mathrm{FC}_{\mathrm{id}}(\boldsymbol{o}_t).$$

5.1.4 Fusion with Long-term User Embedding. A long-term, context-independent embedding is able to explain the general preferences of a user relatively well [22]. We build upon this observation, and enable our RNNs to focus on learning session-specific *deviations* from a long-term user embedding $\bar{\boldsymbol{u}}_t$, defined as a weighted average of all previous session embeddings,

$$\bar{\boldsymbol{u}}_t \propto \sum_{t'=1}^{t-1} \frac{t'}{t-1} \boldsymbol{s}_{t'}, \tag{1}$$

normalized such that $\|\bar{\boldsymbol{u}}_t\| = 1$. To fuse $\bar{\boldsymbol{u}}_t$ and $\hat{\boldsymbol{u}}_t$, we learn attention weights based on the RNN output, such that uncertain RNN estimates can default to the long-term embedding. We compute the attention weights and use those to produce the final user embedding as $\boldsymbol{u}_t = \hat{\beta}_t \hat{\boldsymbol{u}}_t + \bar{\beta}_t \bar{\boldsymbol{u}}_t$, where $\begin{bmatrix} \hat{\beta}_t & \bar{\beta}_t \end{bmatrix} = \mathrm{FC}_{\mathrm{softmax}}(\boldsymbol{o}_t)$.

5.2 Training and Hyperparameter Tuning

To learn the parameters of the model, tune hyperparameters and evaluate the performance of our model, we split the dataset of Section 3 into training, validation and test sets, respectively. The test set consists of all the sessions of the last two weeks of the dataset, and the validation set of the 5 days prior to the beginning of the test set.

5.2.1 Loss function & Optimization. Our model is trained to maximize the cosine similarity between the predicted embedding $\boldsymbol{u}_t$ and the observed one $\boldsymbol{s}_t$. Because both embeddings are unit-norm, the cosine similarity can be computed simply by using a dot product. Formally, letting $\mathcal{D}$ be a training set consisting of pairs (i, t), where i denotes the user and t the session, the loss function is defined as $\ell = \sum_{(i,t) \in \mathcal{D}} \left(1 - \boldsymbol{u}_{it}^\top \boldsymbol{s}_{it}^+\right)$, where $\boldsymbol{u}_{it}$ and $\boldsymbol{s}_{it}^+$ refer to the t-th session of user i. We minimize ℓ using stochastic gradient descent with mini-batches, and find that the Adam optimizer [18] works well, converging in a few tens of epochs.

5.2.2 Hyperparameter Tuning. We use the validation set to tune the learning rate $\lambda \in \{0.001, 0.0005, 0.0001\}$, LSTM cell sizes $d \in \{100, 200, 400\}$, and batch sizes $m \in \{128, 256, 512\}$, and keep fully connected layers fixed to size 200. Optimal performance is achieved by setting $\lambda = 0.0005, d = 400, m = 256$.

6 EXPERIMENTAL EVALUATION

We evaluate the predictive performance of our model on the music sessions dataset described in Section 3. We first present competing approaches (Section 6.1), then we describe two ranking tasks that we use to evaluate our model (Section 6.2) and provide detailed results (Section 6.3). Finally, we perform an ablation study and shed

[5] At $t = 0$, the hidden state is initialized using a learned embedding that depends on the user's age.

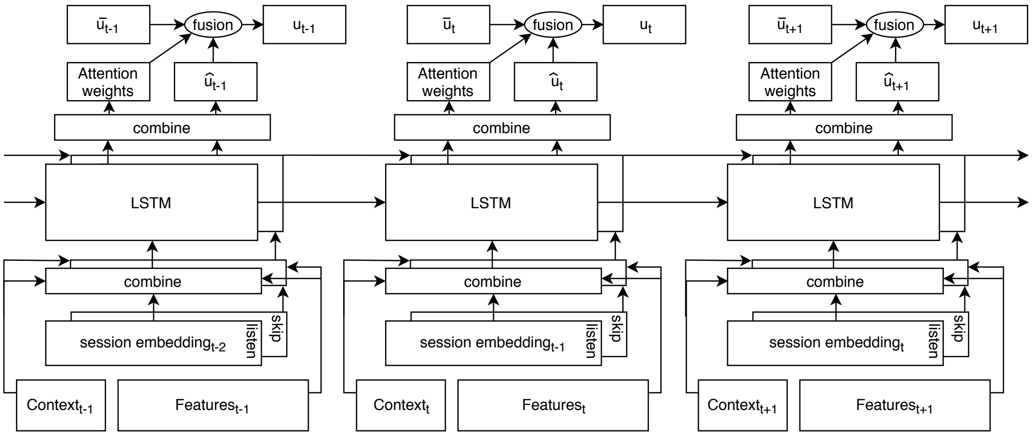


Figure 4: Overview of the CoSeRNN model. The model captures users' sequence-dependent and context-dependent preferences using information available at the beginning of a session.

light on how the different components of our model contribute to its predictive performance (Section 6.4).

6.1 Baselines

We evaluate the performance of several baselines and ground-truth approaches. Our aim is to *a*) understand the various metrics we consider in terms of lower-bounds (achieved by trivial models) and upper bounds (ground-truth), and *b*) tease apart the impact of modelling contextual and sequential effects. We consider the following six baselines.

Last, any cxt This simple baseline predicts the current session embedding using the vector of the last session (irrespective of that session's context), that is, s^{+}_{t-1}. The underlying assumption is that the session vector does not depend on context but that it may change quickly over time.

Last, same cxt Similar to the previous one, except that it uses the vector of the last session whose context is identical to the current one.

Avg, any cxt The current session is modeled as a weighted average of all past session vectors (irrespective of their contexts), similarly to the long-term user embedding in Equation (1).

Avg, same cxt Consists of a weighted average of past sessions as for the previous baseline, except that only sessions with a context that is identical to the current one are considered.

Popularity The predicted current session vector is equal to that of the past session containing the most popular tracks.

JODIE The state-of-the-art embedding prediction model of Kumar et al. [21]. JODIE takes both contextual and sequential effects into account. To match our setup, we leave the track embedding fixed (i.e., we use the existing pretrained embeddings), and use multiple RNNs to represent combinations of all, skipped, and listened parts of the session (similarly to our model). We also use the cosine distance as loss function, as we found that using the ℓ_2-loss (as presented in their paper) obtained inferior results in our setting.

RRN The model of Wu et al. [35]. RRN learns to predict future behavioral trajectories using contextual and sequential information. We tune the parameters used in the original paper and adapt it to our setting by changing the objective function to be the same as that of CoSeRNN (see Section 5.2.1). We replace the original rating prediction layer to have an output of 40 (our embedding size) rather than 1 (their rating score). Similar to JODIE, we leave the embedding of tracks fixed and use multiple RNNs for representing all, skipped, and listened parts of the session.

LatentCross The model of Beutel et al. [3]. LatentCross introduces a method for modulating the state of an RNN model with contextual features. We adapt it to our setting by changing the objective function to be the same as that of CoSeRNN. We also replace the final softmax layer originally used in LatentCross with a feed forward layer of dimension 40 (our embedding size). With this change, LatentCross can be used to generate embedding predictions rather than individual item predictions. Similar to JODIE and RRN, we leave the embedding of tracks fixed, and allow LatentCross to use multiple RNNs for representing all, skipped, and listened parts of the session.

Architectural network choices aside, an important difference between our CoseRNN and the state-of-the-art baselines is in how static (or long-term) user embeddings are combined with recurrent neural network outputs. JODIE uses a static user embedding, RRN learns a stationary user embedding per time step, and LatentCross does not have any. In contrast, CoSeRNN computes a weighted long-term user embedding grounded in a user's actual past consumption, such that the recurrent neural network can focus on learning a sequence and context-dependent *offset* vector, which is fused with the long-term user embedding using attention weights (see Section 5 for further details).

In addition to these baselines, we also examine two oracle variants, to obtain a sense of the difficulty of the various predictive tasks.

Oracle Full The predicted session vector is exactly equal to the (ground-truth) observed one.

Oracle Half The session vector is modeled as the average of half of the tracks of the current session, selected uniformly at random among all the tracks in the session. By using only half of the tracks, it can be seen as a noisy version of Oracle Full, and highlights the inherent variability inside a session.

6.2 Tasks & Metrics

As discussed in Section 5.2, our model is trained to maximize the cosine similarity between the predicted user embedding and observed session embedding. While this loss is attractive from a computational standpoint (being differentiable and smooth), it is merely a useful proxy to the real problem: producing better, more relevant just-in-time recommendations. To assess whether our optimization metric is indeed helping us solving that problem, we evaluate all approaches on two additional tasks.

6.2.1 Session Ranking. The aim here is to measure how well a given approach can discriminate the current session from previous ones. For a given session t, we consider the K session vectors $\{\boldsymbol{s}^+_{t-K+1}, \dots, \boldsymbol{s}^+_t\}$. We rank these session vectors by decreasing cosine similarity with the predicted user embedding $\boldsymbol{u}_t$, and measure the rank of $\boldsymbol{s}_t$. For $K \in \{20, 50\}$, we report the mean reciprocal rank (MRR) and the average rank.

6.2.2 Track Ranking. Here the aim is to measure how well a given approach can predict the tracks that a user listens to in a given session. Similarly to session ranking, this measures if the approaches are able to adapt to the user's current preferences, but where the individual items are considered in contrast to an aggregated session representation. Given a session t, we consider the set of K distinct tracks a user has listened to most recently (across all previous sessions). For $K \in \{25, 100\}$, we rank these tracks by decreasing cosine similarity with the predicted vector $\hat{\boldsymbol{s}}_t$ and report the mean average precision (mAP) and the average recall@10. If we are able to rank tracks that are contained in the session highly, it means that we can anticipate the user behavior well, e.g., by showing the relevant tracks more prominently (or even start playing them directly).

Experimental Setup. We use the dataset described in Section 3 and the training procedure of Section 5.2. For all methods, when predicting the user embedding of the t-th session, we use all the data up to (but not including) session t.

6.3 Results

Table 2 presents the performance on the test set for the cosine-similarity loss as well as for the various metrics used for the session and track ranking tasks. *CoSeRNN* consistently outperforms the baselines on all metrics across all tasks (all differences are statistically significant using a pairwise two-tailed t-test at the 0.001 level). It is interesting to note that although there is a clear positive correlation between cosine similarity and the other metrics, a higher cosine similarity does not automatically imply better performance on the ranking tasks.

The four models that are optimized on the cosine similarity (*RRN*, *LatentCross*, *JODIE*, and *CoSeRNN*) are clearly superior to the simple heuristic baselines on both ranking tasks. Among all simple baselines, it is worth noting that *Last, any cxt* generally performs the best (except on cosine and on session ranking for $K = 20$), an indication that recency plays an important role in music recommendation; we investigate this further in Section 6.4. This also explains part of the performance gap from the simple baselines to JODIE and CoSeRNN, as they are able to better capture the recency aspect through recurrent neural networks.

The results up to now are averages over all users and test sessions. We now seek to answer the question: does the performance vary across contexts? We plot the relative improvement of *CoSeRNN* over *JODIE* (the state-of-the-art approach) in Figure 5. For conciseness, we only consider a subset of the metrics, and use $K = 50$ and $K = 100$ for the session and track ranking tasks, respectively. Generally, the improvements are consistent across all contexts. Interestingly, some of the contexts that occur infrequently see a comparatively larger relative improvement, such as for the *web* or—to a smaller extent—*night* contexts.

6.4 Ablation Study

We focus on the empirical performance of *CoSeRNN* and seek to understand how different choices affects the model, which we ablate in two different ways: 1) by varying the features given as input to the model, and 2) by processing played and skipped tracks in different ways.

6.4.1 Input Features. We divide features given as input to our model into five groups: the embeddings of the last session $\boldsymbol{s}^+_t$ and $\boldsymbol{s}^-_t$, the current context $\boldsymbol{c}_t$, the number of tracks in the last session N_{t-1}, the time (in seconds) elapsed since the last session Δ_t, and the stream source of the last session z_{t-1}. Additionally, we consider a hypothetical scenario[6] where we have access to the stream source of the *current* session, z_t.

The performance of the corresponding models on the cosine similarity and track ranking tasks is given in Table 3. The current session context is associated with the biggest increase in cosine similarity, which is to be expected as context is highly indicative of the content in a session (as seen in Section 4). In addition, information about the previous session significantly helps improving performance, particularly on the track ranking task. If, in addition to knowing a user device and the time of the day, we know which stream source they intend to stream from, our model predictive performance increases substantially.

6.4.2 Listened vs. Skipped Tracks. The final architecture of our model partitions tracks in a session into two subsets, *played* tracks and *skipped* tracks. To better understand the impact of this particular choice, we compare our model to three alternative variants. The first one does not distinguish between played and skipped tracks, and instead considers the average embedding of *all* tracks in the session. The second one disregards skipped tracks completely and focuses only on played tracks. The last one considers played and skipped tracks separately, but also adds in another RNN pathway that considers the average of all tracks. Table 4 displays the performance attained by each model.

[6] This scenario assumes that we are given partial information about the user intent in the current session. This assumption is realistic in practice, but the trade-off is that we cannot present recommendations immediately at app launch time.

Table 2: Empirical performance of various session-prediction approaches on a music dataset. Best result is highlighted in bold (all differences are statistically significant at the 0.001 level using a paired two-tailed t-test).

		Session ranking				Track ranking			
		K = 20		K = 50		K = 25		K = 100	
Model	Cosine	MRR	Rank	MRR	Rank	mAP	Rec@10	mAP	Rec@10
Last, any cxt	0.6527	0.1721	10.0767	0.1133	21.9245	0.3585	0.4394	0.1300	0.1372
Last, same cxt	0.5990	0.2094	9.3128	0.1009	23.5288	0.3398	0.4223	0.1154	0.1156
Avg, any cxt	0.6797	0.1835	10.0882	0.1034	23.7337	0.3485	0.4250	0.1225	0.1291
Avg, same cxt	0.6609	0.2087	9.3365	0.1031	23.4767	0.3476	0.4313	0.1199	0.1239
Popularity	0.4278	0.2012	9.5670	0.0967	24.0233	0.3457	0.4338	0.1165	0.1190
RRN [35]	0.6918	0.1970	9.6689	0.1286	21.2980	0.3794	0.4678	0.1425	0.1594
LatentCross [3]	0.6921	0.1967	9.6669	0.1286	21.2610	0.3794	0.4678	0.1422	0.1592
JODIE [21]	0.6970	0.2079	9.3438	0.1303	21.2258	0.3836	0.4734	0.1450	0.1638
CoSeRNN (ours)	**0.7115**	**0.2319**	**8.6642**	**0.1507**	**19.5288**	**0.4011**	**0.4981**	**0.1574**	**0.1816**
Oracle half	0.7077	0.4723	5.2294	0.3711	11.3131	0.7732	0.8052	0.5824	0.6163
Oracle full	1.0000	0.9985	1.0067	0.9988	1.0068	0.8323	0.8861	0.6220	0.6951

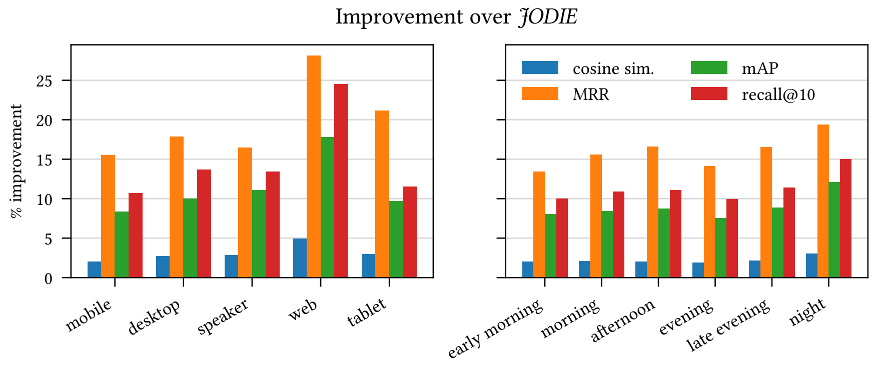


Figure 5: Improvement of *CoSeRNN* over *JODIE*. We compute MRR for $K = 50$ and mAP and recall@10 for $K = 100$.

Table 3: Performance of *CoSeRNN* model variants with access to increasing subsets of input features.

		Track ranking (K = 100)	
Features	Cosine	mAP	Recall@10
Last sess. embeddings	0.7062	0.1518	0.1731
+ curr. context	0.7091	0.1527	0.1748
+ # tracks in last	0.7103	0.1569	0.1807
+ time since last	0.7109	0.1569	0.1808
+ last stream source	0.7115	0.1574	0.1816
+ curr. stream source	**0.7313**	**0.1777**	**0.2100**

Table 4: Performance of four *CoSeRNN* model variants that encode the session in different ways.

		Track ranking (K = 100)	
Session encoding	Cosine	mAP	Recall@10
All	0.6966	0.1442	0.1623
Plays	0.7103	0.1567	0.1801
Plays + skips	**0.7115**	**0.1574**	**0.1816**
Plays + skips + all	0.7114	0.1569	0.1809

Separating played tracks from skipped tracks is clearly beneficial to model performance. Interestingly, considering skipped tracks in addition to played tracks does bring some performance benefits. Even though the performance benefit is small, this does highlight the dissimilarities between a user skipped and listened tracks, and it helps to improve the model capabilities of understand a user music preferences. Finally, considering the union of played and skipped tracks in addition to the two partitions is unnecessary and does not improve performance.

7 CONCLUSION

In this work, we consider the task of learning contextual and sequential user embeddings suited for music recommendation at the beginning of a session. To this end, we first perform multiple exploratory analyses, gaining a better understanding of how sessions are distributed according to context, how music consumption varies depending on context, and how context correlates with the tracks within a session. We find that most users experience a diversity of contexts (even though some occur more frequently than others), that sessions belonging to rarely occurring contexts vary the most (in terms of contents), and that sessions with the same context have more similar content.

Driven by these findings, we present CoSeRNN, a recurrent neural network embedding model that learns the sequential listening behaviour of users, and adapts it to the current context. CoSeRNN does this through the combination of *a*) a global long-term embedding that captures a user's long-term music preferences, and *b*) a sequence and context-dependent offset. In contrast to prior methods that require expensive model evaluations to produce recommendations, the approach taken by CoSeRNN enables efficiently generating recommendations by using fast approximate nearest neighbour searches. When evaluated empirically on a large-scale dataset of sessions, CoSeRNN outperforms baseline and state-of-the-art embedding-based approaches by upwards of 10% in session and track recommendation tasks. In future work, hashing-based embedding approaches would be interesting to investigate in our setting, as existing work on content-aware recommendation [12, 39] and similarity search [11, 13, 32] have shown hashing-based approaches to allow large efficiency gains at the cost of a marginal effectiveness reduction.

REFERENCES

[1] A. Anderson, R. Kumar, A. Tomkins, and S. Vassilvitskii. 2014. The dynamics of repeat consumption. In *international conference on World wide web*. ACM, 419–430.

[2] Y. Bachrach, Y. Finkelstein, R. Gilad-Bachrach, L. Katzir, N. Koenigstein, N. Nice, and U. Paquet. 2014. Speeding up the xbox recommender system using a euclidean transformation for inner-product spaces. In *Conference on Recommender systems*. ACM, 257–264.

[3] A. Beutel, P. Covington, S. Jain, C. Xu, J. Li, V. Gatto, and E. H. Chi. 2018. Latent cross: Making use of context in recurrent recommender systems. In *International Conference on Web Search and Data Mining*. 46–54.

[4] G. Bonnin and D. Jannach. 2014. Automated generation of music playlists: Survey and experiments. *ACM Computing Surveys (CSUR)* 47, 2 (2014), 1–35.

[5] B. Brost, R. Mehrotra, and T. Jehan. 2019. The Music Streaming Sessions Dataset. In *The World Wide Web Conference*. ACM, 2594–2600.

[6] T. Cebrián, M. Planagumà, P. Villegas, and X. Amatriain. 2010. Music recommendations with temporal context awareness. In *Conference on Recommender systems*. ACM, 349–352.

[7] J. Chen, C. Wang, and J. Wang. 2015. Will you" reconsume" the near past? fast prediction on short-term reconsumption behaviors. In *Conference on Artificial Intelligence*.

[8] S. Chen, J. L. Moore, D. Turnbull, and T. Joachims. 2012. Playlist prediction via metric embedding. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*. 714–722.

[9] H. Dai, Y. Wang, R. Trivedi, and L. Song. 2016. Deep coevolutionary network: Embedding user and item features for recommendation. *arXiv preprint arXiv:1609.03675* (2016).

[10] E. Frolov and I. Oseledets. 2017. Tensor methods and recommender systems. *Interdisciplinary Reviews: DataMining and Knowledge Discovery* (2017).

[11] C. Hansen, C. Hansen, J. G. Simonsen, S. Alstrup, and C. Lioma. 2019. Unsupervised Neural Generative Semantic Hashing. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 735–744.

[12] C. Hansen, C. Hansen, J. G. Simonsen, S. Alstrup, and C. Lioma. 2020. Content-Aware Neural Hashing for Cold-Start Recommendation. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 971–980.

[13] C. Hansen, C. Hansen, J. G. Simonsen, S. Alstrup, and C. Lioma. 2020. Unsupervised Semantic Hashing with Pairwise Reconstruction. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2009–2012.

[14] N. Hariri, B. Mobasher, and R. Burke. 2012. Context-aware music recommendation based on latenttopic sequential patterns. In *Proceedings of the sixth ACM conference on Recommender systems*. 131–138.

[15] B. Hidasi, A. Karatzoglou, L. Baltrunas, and D. Tikk. 2016. Session-based recommendations with recurrent neural networks. (2016).

[16] S. Hochreiter and J. Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

[17] M. Kaminskas, F. Ricci, and M. Schedl. 2013. Location-aware music recommendation using auto-tagging and hybrid matching. In *Conference on Recommender systems*. ACM, 17–24.

[18] D. P. Kingma and J. Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[19] Y. J. Ko, L. Maystre, and M. Grossglauser. 2016. Collaborative Recurrent Neural Networks for Dynamic Recommender Systems. In *ACML*, Vol. 63.

[20] Y. Koren, R. Bell, and C. Volinsky. 2009. Matrix Factorization Techniques for Recommender Systems. *Computer* 42, 8 (Aug. 2009), 30–37.

[21] S. Kumar, X. Zhang, and J. Leskovec. 2019. Predicting dynamic embedding trajectory in temporal interaction networks. In *International Conference on Knowledge Discovery & Data Mining*. ACM, 1269–1278.

[22] R. Mehrotra, J. McInerney, H. Bouchard, M. Lalmas, and F. Diaz. 2018. Towards a fair marketplace: Counterfactual evaluation of the trade-off between relevance, fairness & satisfaction in recommendation systems. In *International Conference on Information and Knowledge Management*. ACM, 2243–2251.

[23] T. Mikolov, K. Chen, G. Corrado, and J. Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* (2013).

[24] M. Park, J. Thom, S. Mennicken, H. Cramer, and M. Macy. 2019. Global music streaming data reveal diurnal and seasonal patterns of affective preference. *Nature Human Behaviour* 3, 3 (2019), 230.

[25] M. J. Pazzani and D. Billsus. 2007. Content-based recommendation systems. In *The adaptive web*. Springer, 325–341.

[26] T. F. Pettijohn, G. M. Williams, and T. C. Carter. 2010. Music for the seasons: seasonal music preferences in college students. *Current Psychology* 29, 4 (2010), 328–345.

[27] M. Quadrana, P. Cremonesi, and D. Jannach. 2018. Sequence-aware recommender systems. *ACM Computing Surveys (CSUR)* 51, 4 (2018), 1–36.

[28] S. Raza and C. Ding. 2019. Progress in context-aware recommender systems—an overview. *Computer Science Review* 31 (2019), 84–97.

[29] P. Ren, Z. Chen, J. Li, Z. Ren, J. Ma, and M. de Rijke. 2019. RepeatNet: A repeat aware neural recommendation machine for session-based recommendation. In *Conference on Artificial Intelligence*, Vol. 33. 4806–4813.

[30] M. Schedl, H. Zamani, C.-W. Chen, Y. Deldjoo, and M. Elahi. 2018. Current challenges and visions in music recommender systems research. *Journal of Multimedia Information Retrieval* (2018), 95–116.

[31] G. Shani, D. Heckerman, and R. I. Brafman. 2005. An MDP-based recommender system. *Journal of Machine Learning Research* (2005), 1265–1295.

[32] D. Shen, Q. Su, P. Chapfuwa, W. Wang, G. Wang, R. Henao, and L. Carin. 2018. NASH: Toward End-to-End Neural Architecture for Generative Semantic Hashing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 2041–2050.

[33] A. Vall, M. Quadrana, M. Schedl, G. Widmer, and P. Cremonesi. 2017. The Importance of Song Context in Music Playlists.. In *RecSys Posters*.

[34] S. Wang, L. Cao, and Y. Wang. 2019. A survey on session-based recommender systems. *arXiv preprint arXiv:1902.04864* (2019).

[35] C.-Y. Wu, A. Ahmed, A. Beutel, A. J. Smola, and H. Jing. 2017. Recurrent recommender networks. In *International conference on web search and data mining*. ACM, 495–503.

[36] S. Zhang, Y. Tay, L. Yao, and A. Sun. 2018. Next Item Recommendation with Self-Attention. *ArXiv* abs/1808.06414 (2018).

[37] S. Zhang, L. Yao, A. Sun, and Y. Tay. 2019. Deep Learning Based Recommender System: A Survey and New Perspectives. *ACM Comput. Surv.* 52, 1 (2019).

[38] Y. Zhang, H. Dai, C. Xu, J. Feng, T. Wang, J. Bian, B. Wang, and T.-Y. Liu. 2014. Sequential click prediction for sponsored search with recurrent neural networks. In *Conference on Artificial Intelligence*.

[39] Y. Zhang, H. Yin, Z. Huang, X. Du, G. Yang, and D. Lian. 2018. Discrete Deep Learning for Fast Content-Aware Recommendation. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*. 717–726.

[40] A. Zimdars, D. M. Chickering, and C. Meek. 2001. Using temporal data for making recommendations. In *Conference on Uncertainty in artificial intelligence*. Morgan Kaufmann Publishers Inc., 580–588.

Chapter 4

Modelling Sequential Music Track Skips using a Multi-RNN Approach

Christian Hansen, Casper Hansen, Jakob Grue Simonsen, Stephen Alstrup, Christina Lioma (2019). Modelling Sequential Music Track Skips Using a Multi-RNN Approach. In WSDM Cup. [42].

Modelling Sequential Music Track Skips using a Multi-RNN Approach

Christian Hansen
University of Copenhagen
Department of Computer Science
chrh@di.ku.dk

Casper Hansen
University of Copenhagen
Department of Computer Science
c.hansen@di.ku.dk

Stephen Alstrup
University of Copenhagen
Department of Computer Science
s.alstrup@di.ku.dk

Jakob Grue Simonsen
University of Copenhagen
Department of Computer Science
simonsen@di.ku.dk

Christina Lioma
University of Copenhagen
Department of Computer Science
c.lioma@di.ku.dk

ABSTRACT

Modelling sequential music skips provides streaming companies the ability to better understand the needs of the user base, resulting in a better user experience by reducing the need to manually skip certain music tracks. This paper describes the solution of the University of Copenhagen ”DIKU-IR” team in the ”Spotify Sequential Skip Prediction Challenge”, where the task was to predict the skip behaviour of the second half in a music listening session conditioned on the first half. We model this task using a Multi-RNN approach consisting of two distinct stacked recurrent neural networks, where one network focuses on encoding the first half of the session and the other network focuses on utilizing the encoding to make sequential skip predictions. The encoder network is initialized by a learned session-wide music encoding, and both of them utilize a learned track embedding. Our final model consists of a majority voted ensemble of individually trained models, and ranked 2nd out of 45 participating teams in the competition with a mean average accuracy of 0.641 and an accuracy on the first skip prediction of 0.807. Our code is released at https://github.com/Varyn/WSDM-challenge-2019-spotify.

KEYWORDS

Music Track Recommendation, Skip Prediction, Music Embedding, Recurrent Neural Network, Deep Learning

ACM Reference Format:
Christian Hansen, Casper Hansen, Stephen Alstrup, Jakob Grue Simonsen, and Christina Lioma. 2018. Modelling Sequential Music Track Skips using a Multi-RNN Approach. In *Proceedings of The 12th ACM International Conference on Web Search and Data Mining (WSDM'19).* ACM, New York, NY, USA, Article 4, 4 pages. https://doi.org/10.475/123_4

1 INTRODUCTION

A challenge for content providers is to model how a given user will react to some content, such that users are provided with content that elicits some positive reaction. Spotify, a music streaming company, have the problem of incorporating the sequential nature of a listening session to predict whether a user will skip a given music track or not[1]. To this end, a set of approximately 130 million labelled user sessions has been released, where each session consists of 10 to 20 playback tracks. The task is to predict which music tracks a user will skip in the second half of session conditioned on the first half. This problem can be considered a type of session based recommendation, since no explicit user profile is available, and the user preferences should therefore be estimated within the first half of the session. To capture the dynamics of the session, models based on recurrent neural networks (RNNs) have seen much popularity for session based recommender systems, as they are able to encode temporal information well [5, 8, 10].

In this paper we present our solution to the "Spotify Sequential Skip Prediction Challenge", which ranked second among all 45 submitted solutions. Our solution is based on using two distinct stacked RNNs, one stacked RNN to encode the first half of a session, and one stacked RNN responsible for making the skip predictions conditioned on the state of the first RNN. The stacked RNN used for encoding the first half of the session is initialized using meta features related to the session as a whole, and an encoding of all tracks listened to in the entire session. Both RNNs utilize a track embedding based on provided track features and a learned embedding. Our model shares similarities with architectures used in sequence-to-sequence networks [9], which also consist of two distinct RNNs for encoding and decoding. Sequence-to-sequence networks are especially popular for machine translation where a strong encoding of the sentence is needed [4, 11].

2 SEQUENTIAL MUSIC SKIP PREDICTION

In this section we present our method. First we present the Spotify dataset [3] and its features for both the sessions and the tracks. We thereafter present an overview of the model, and then go into greater detail of each component of the model.

2.1 Features

We first describe the dataset and features to establish a common terminology to use throughout the paper, and then the feature

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
WSDM'19, February, 2019, Australia
© 2018 Copyright held by the owner/author(s).
ACM ISBN 123-4567-24-567/08/06.
https://doi.org/10.475/123_4

[1] https://www.crowdai.org/challenges/spotify-sequential-skip-prediction-challenge

engineering used to process the data. The dataset consists of two primary parts consisting of a user log and a track dataset.

The user log contains the user sessions, which for each user is a sequence of track playbacks. A track playback contains information about the user in general (e.g., if they are premium or the day of the week they are listening), features related to a single track playback (e.g., what action the user took to end up listening to the current track, or which action the user took to stop listening to this track), and lastly the id of the track being listened to. The training data contains the playback track features for all tracks in the user session. For the testing data this is available for the first half of the session, while the last half of the session only contains a track id and position in the session of the track being played.

The track dataset contains a number of features for each track which both relate to meta information about the song (e.g., popularity and release information) and specific information related to the musical content of the song (e.g., beat strength or flatness).

2.1.1 Feature processing. Our method is not reliant on any extensive data pre-processing, so the data preparation is straight forward: the user session is represented as 3 types of data:

(1) Meta information associated with the whole session
(2) A sequence of playback tracks for the first half of the listening session
(3) A sequence of track ids and position in the overall session for the second half of the listening session. This was done to mimic how the testing data was constructed.

The meta information (1) for the whole session consists of whether the user is a premium user, the length of the session, and the day of the week. These are encoded separately using a one hot encoding. The first half of the listening session (2) contains all the features for each playback track, except for the features listed in the meta information (1). All categorical features are encoded using a one hot encoding. The second half of the listening session (3) only contains the track id and position in the session for each track.

The track data is standardized such that each feature has 0 mean and unit variance, and is otherwise used as is for representing a track.

2.2 Model

This section will explain the neural architecture of our model, and the network can be seen in Figure 1. Overall, the network consists of 4 parts: 1) An embedding of the tracks; 2) A network for encoding all tracks in the full session; 3) A network for encoding the playback track sequence, which is the first half of a session; and lastly 4) A prediction network that takes the encoding from (3) and makes a skip prediction for each track of the second half of the session.

2.2.1 Track embedding. The purpose of the track embedding is to produce an embedding of the track that both utilizes the provided features, but also allows the network to learn an embedding specifically optimized for the skip prediction task. These two kinds of embeddings are concatenated into a single embedding.

The track embedding has 2 tunable parameters: the size of the learned embedding for the track and the size of the final track embedding. Each track has an identifier, i, which is used to index into two embedding matrices E_{fixed} and $E_{learned}$. E_{fixed} contains all features from the track data, which have been normalized as described in Section 2.1. $E_{learned}$ is a learned embedding, which is initialized using a uniform distribution with values from $[-0.05, 0.05]$. The final track embedding is then computed as:

$$\mathbf{track}_i = \mathrm{ReLU}\big(W_t[E_{\text{fixed}}\mathbf{w}_i \oplus E_{\text{learned}}\mathbf{w}_i] + \mathbf{b}_t\big) \tag{1}$$

where ReLU is the Rectified Linear Unit activation function, $\oplus$ corresponds to vector concatenation, and $\mathbf{w}_i$ is the identity vector whose ith entry is 1 and all other entries 0. The size of $\mathbf{track}_i$ is the final track embedding size.

2.2.2 Session encoding. The purpose of making an initial encoding of the whole session (consisting of all tracks), is to allow the network to be able to learn what kind of music is listened to across the session. The goal of this is to allow the network to utilize variation, genre, and general music similarity throughout the session. To do this, the track embedding presented in the previous section is used. For a session we have a sequence of embedded tracks $[\mathbf{track}_{i_1}, \mathbf{track}_{i_2}, ..., \mathbf{track}_{i_m}]$, where i_1 is the track embedding of the first track of the session and m is the length of the session. We first apply a RNN with Long Short-Term Memory (LSTM) units [6], and get the output produced for each timestep:

$$[\mathbf{o}_1, \mathbf{o}_2, ..., \mathbf{o}_m] = \mathrm{LSTM}([\mathbf{track}_{i_1}, \mathbf{track}_{i_2}, ..., \mathbf{track}_{i_m}]) \tag{2}$$

where $\mathbf{o}_i$ is the ith output of the LSTM, and $\mathrm{LSTM}(\cdot)$ is the application of the LSTM on the whole sequence. The final session embedding is made using an attention-weighted sum over the outputs,

$$\mathbf{session} = \sum_{i=1}^{m} \mathbf{o}_i \frac{\exp(W_a\mathbf{o}_i + b_a)}{\sum_{j=1}^{m}\exp(W_a\mathbf{o}_j + b_a)} \tag{3}$$

where the second term of the sum corresponds to the attention weighting of the ith output, exp is the exponential function, and **session** is the session embedding. The size of the session embedding is dependent on the size of the LSTM unit.

2.2.3 Playback encoding. The purpose of the playback encoding network is to make an encoding of the first half of the session, and then use this encoding when making the skip prediction for each track in the second half of the session.

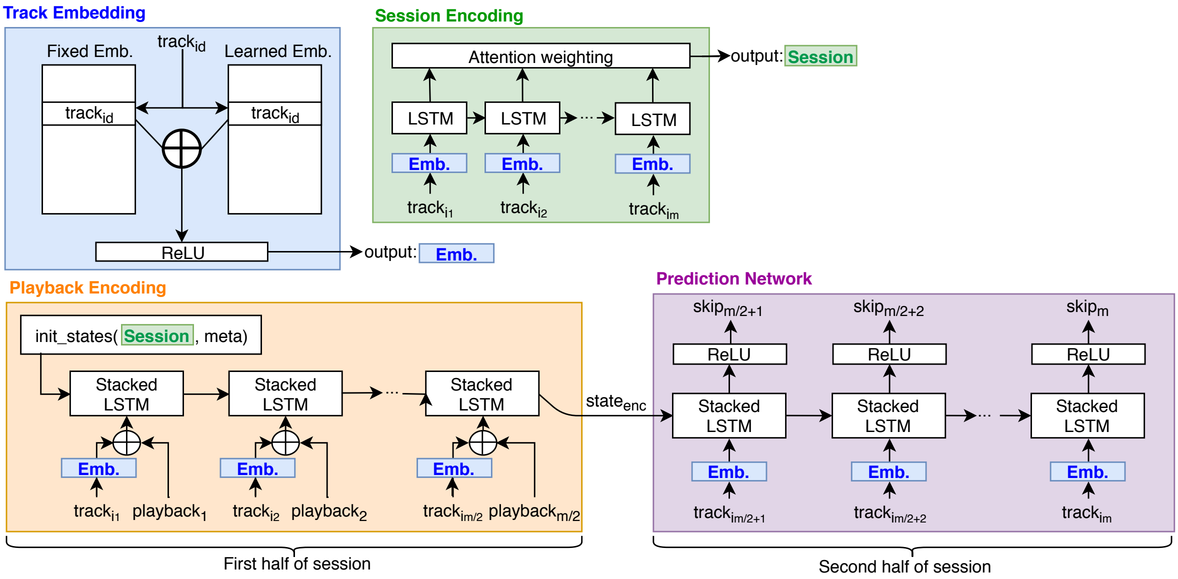
The encoding network uses a stacked RNN with a depth of 2 using LSTM units. The initial state of the LSTM is given by 4 linear fully connected layers taking as input the encoding of all tracks and the session meta information, and producing the initial hidden state and output of the LSTM, which we refer to as hidden and output respectively:

$$\mathbf{hidden}_l^{\text{initial}} = W_{s,l}[\mathbf{m} \oplus \mathbf{session}] + \mathbf{b}_{s,l} \tag{4}$$

$$\mathbf{output}_l^{\text{initial}} = W_{o,l}[\mathbf{m} \oplus \mathbf{session}] + \mathbf{b}_{o,l} \tag{5}$$

where $l \in \{1, 2\}$ indicates the first or second LSTM layer. For later use we will simply refer to the initial state of the whole stacked LSTM as $\text{state}^{\text{initial}}$. $\mathbf{m}$ is the meta information associated with the session as described in Section 2.1. The input to the stacked LSTM at the jth timestep for the first half of the session, $\mathbf{s}_j^{\text{f}}$, is constructed as:

$$\mathbf{s}_j^{\text{f}} = [\mathbf{track}_{i_j} \oplus \mathbf{playback}_j] \tag{6}$$



Track Embedding
Fixed Emb.
track_{id}
Learned Emb.
track_{id}
track_{id}
ReLU
output: Emb.
Session Encoding
Attention weighting
output: Session
LSTM
LSTM
LSTM
Emb.
Emb.
Emb.
track_{i1}
track_{i2}
track_{im}
Playback Encoding
init_states(Session , meta)
Stacked LSTM
Stacked LSTM
Stacked LSTM
Emb.
Emb.
Emb.
track_{i1} playback_1
track_{i2} playback_2
$\text{track}_{\text{im/2}}$ $\text{playback}_{\text{m/2}}$
$\text{state}_{\text{enc}}$
Prediction Network
$\text{skip}_{\text{m/2+1}}$
$\text{skip}_{\text{m/2+2}}$
skip_{m}
ReLU
ReLU
ReLU
Stacked LSTM
Stacked LSTM
Stacked LSTM
Emb.
Emb.
Emb.
$\text{track}_{\text{im/2+1}}$
$\text{track}_{\text{im/2+2}}$
track_{im}
First half of session
Second half of session

Figure 1: Network architecture. ⊕ represents vector concatenation of the two embedding look-ups in the Track Embedding and between an embedded track and playback information in the Playback Encoder.

where $\mathbf{playback}_j$ is the playback track features described in Section 2.1, for the jth track in the session. The sequence for the first half is then $[\mathbf{s}^{\text{f}}_1, \mathbf{s}^{\text{f}}_2, ..., \mathbf{s}^{\text{f}}_{\frac{m}{2}}]$. The playback encoding is the final state of the LSTM after the whole sequence has been read:

$$\text{state}_{\text{enc}} = \text{STACK-LSTM}_{\text{enc}}([\mathbf{s}^{\text{f}}_1, \mathbf{s}^{\text{f}}_2, ..., \mathbf{s}^{\text{f}}_{\frac{m}{2}}] | \text{state}^{\text{initial}}) \quad (7)$$

where $\text{state}_{\text{enc}}$ is the final state of the stacked LSTM, and $\text{STACK-LSTM}_{\text{enc}}(\cdot)$ is the application of the stacked LSTM on the input sequence conditioned on the initial state, $\text{state}^{\text{initial}}$. The size of the LSTMs in both layers are chosen to be the same.

2.2.4 Prediction network. The purpose of the prediction network is to make a prediction for each track in the second half of the session. This is done by using a stacked LSTM that reads the second half of the session and produces an output for each track, which is used to make the final predictions. Note that the stacked LSTMs in the encoder and the prediction network have the same LSTM size, but do not share any weights. The input at timestep j for the stacked LSTM in the prediction network is a track embedding concatenated with the position of the track in the full session ($j + \frac{m}{2}$). We denote the input sequence to the stacked LSTM as: $[\mathbf{s}^s_1, \mathbf{s}^s_2, ..., \mathbf{s}^s_{\frac{m}{2}}]$ and compute the stacked LSTM as,

$$[\mathbf{o}^s_1, \mathbf{o}^s_2, ..., \mathbf{o}^s_{\frac{m}{2}}] = \text{STACK-LSTM}_{\text{pred}}([\mathbf{s}^s_1, \mathbf{s}^s_2, ..., \mathbf{s}^s_{\frac{m}{2}}] | \text{state}_{\text{enc}}) \quad (8)$$

The skip prediction for each track is then given as the output of two fully connected layers, where the first layer has the same size as the LSTM and a ReLU activation function, whereas the second layer produces the prediction via a sigmoid activation function.

The whole network is trained using binary cross entropy, with the ground truth skip behavior of the user as the target for each prediction.

3 EXPERIMENTAL EVALUATION

We now describe the experimental evaluation of our model. We first describe the performance metric and simple baselines provided by the competition organizers, followed by how our model was tuned as well as implementation details. We then study different configurations of our model on a validation set, and lastly report the test performance of our final model.

3.1 Performance metric

The competition employs the Mean Average Accuracy as the official performance metric. The average accuracy is computed for each session and then averaged across all sessions. The average accuracy is defined as:

$$AA = \frac{\sum_{i=1}^{T} A(i)L(i)}{T} \quad (9)$$

where T is the number of tracks to be predicted in a session, $A(i)$ is the accuracy at position i of the track sequence, and $L(i) = 1$ if the prediction at position i was correct, and $L(i) = 0$ otherwise. Additionally, for the test performance we also report the accuracy on the first skip prediction, as that was used for breaking potential ties among the participants.

3.2 Baselines

For comparison we include the 3 provided baselines by the competition organizers; the baselines are relatively simple, but show the performance using intuitive rules for making the skip decisions:

(1) Predict all tracks to be skipped;
(2) Predict track to be skipped if its skip rate in the training set was greater than 0.5;
(3) Predict the last action from first the half of the session for all tracks in the second half of the session.

We report the performance metrics on these baselines for the final comparison on the test set and refer to them as Baseline 1-3.

3.3 Tuning

We measure the effectiveness of our model using mean average accuracy and maximize this metric for tuning parameters. For training we randomly shuffle the provided dataset of 130 million sessions, and set aside 0.2 million sessions for validation, which will be used to detect overfitting. Testing is done on a separate dataset consisting of 31.3 million sessions.

Due to the massive size of the dataset we fix the network parameters with a track embedding of size 50; a fully connected layer with size 350 for combining the fixed track embedding with the learned track embedding; a LSTM size of 100 for the encoding of tracks in a session; an LSTM size of 500 in both the encoder and predictor networks; and lastly a fully connected layer with size 500 for making the skip prediction of the predictor network output. These were initially decided based on rapid tests trained for a short number of iterations, but due to computational limitations we did not explore them in full detail. For the network training parameters we explored batch sizes in the set $\{50, 100, 200, 300, 400\}$, and used a learning rate of 0.0005. For training the network we used the Adam optimizer [7] with a learning rate of 0.0005. We trained the models using Titan X GPUs, and used 40-60 hours to fully train each model.

3.4 Implementation details

Due to the massive size of the dataset we implemented the LSTMs in our model using `cuDNNLSTM` in TensorFlow [1], which is up to 7.2 times faster than traditional LSTM implementations [2]. However, this implementation requires a fixed amount of time steps, which was set to 10 for both RNNs. To handle sessions of varying length we applied pre-padding on the input to the playback encoder network and post-padding to the input to the prediction network, such that all inputs fitted the fixed size.

3.5 Results

Table 1 displays the validation mean average accuracy of our model when varying the batch size, which we found to be one of the most influential parameters to tune for our network architecture. The table shows that a larger batch size leads to a higher average accuracy with a batch size of 300 performing the best.

Table 2 shows the test mean average accuracy and accuracy of the first skip prediction. The table shows the performance of the three provided baselines, our best performing submission of a single model (using a batch size of 300), as well as a majority voting among the 5 models with varying batch size. We observed that the average correlation between the predictions made by the 5 models was 0.851, and created a final model by taking a majority voting among the models. The majority voted model performed better than the best single model with an average accuracy of 0.641 and first skip prediction accuracy of 0.807, which was also notably better than the three baselines. On the final leaderboard this ranked as the second best submission in the competition.

Batch size	Validation AA
400	0.637
300	**0.638**
200	0.635
100	0.633
50	0.631

Table 1: Model validation performance

Model	Test AA	First Prediction Accuracy
Baseline 1	0.405	0.541
Baseline 2	0.409	0.559
Baseline 3	0.537	0.742
Our model	0.638	0.805
Our majority voted model	**0.641**	**0.807**

Table 2: Model test performance

4 CONCLUSION

This paper described the participation of the University of Copenhagen ”DIKU-IR” team in the ”Spotify Sequential Skip Prediction Challenge”, where the task was to predict the skip behaviour of the second half in a music listening session conditioned on the first half. We proposed a new model for the task of modelling sequential music skip behaviour in a given user session of streamed music content. Our model consisted of a Multi-RNN approach with two distinct stacked recurrent neural networks, which can be considered as an encoding and predictor network. The encoder network exploited a learned session-wide encoding of the musical content, while both of them utilized a learned embedding of each musical track. We combined individually trained models with different batch sizes in a majority voted ensemble to obtain a mean average accuracy of 0.641 and an accuracy on the first skip prediction of 0.807, which was the second best submission of the competition among 45 participating teams.

ACKNOWLEDGMENTS

Funded by the Innovation Fund Denmark, DABAI project.

REFERENCES

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. 2016. Tensorflow: a system for large-scale machine learning.. In *OSDI*, Vol. 16. 265–283.

[2] Stefan Braun. 2018. LSTM Benchmarks for Deep Learning Frameworks. (2018). http://arxiv.org/abs/1806.01818

[3] Brian Brost, Rishabh Mehrotra, and Tristan Jehan. 2019. The Music Streaming Sessions Dataset. In *Proceedings of the 2019 Web Conference*. ACM.

[4] Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. 2017. Convolutional Sequence to Sequence Learning. In *International Conference on Machine Learning*. 1243–1252.

[5] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2015. Session-based recommendations with recurrent neural networks. *arXiv preprint arXiv:1511.06939* (2015).

[6] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

[7] Diederik P. Kingma and Jimmy Ba. 2015. A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*.

[8] Massimo Quadrana, Alexandros Karatzoglou, Balázs Hidasi, and Paolo Cremonesi. 2017. Personalizing session-based recommendations with hierarchical recurrent neural networks. In *Proceedings of the Eleventh ACM Conference on Recommender Systems*. ACM, 130–137.

[9] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*. 3104–3112.

[10] Yong Kiam Tan, Xinxing Xu, and Yong Liu. 2016. Improved recurrent neural networks for session-based recommendations. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*. ACM, 17–22.

[11] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, et al. 2016. Google’s neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144* (2016).

Chapter 5

MultiFC: A Real-World Multi-Domain Dataset for Evidence-Based Fact Checking of Claims

Isabelle Augenstein, Christina Lioma, Dongsheng Wang, Lucas Chaves Lima, Casper Hansen, Christian Hansen, Jakob Grue Simonsen (2019). MultiFC: A Real-World Multi-Domain Dataset for Evidence-Based Fact Checking of Claims. In EMNLP, pages 4685-4697. [7].

MultiFC: A Real-World Multi-Domain Dataset for Evidence-Based Fact Checking of Claims

Isabelle Augenstein **Christina Lioma** **Dongsheng Wang** **Lucas Chaves Lima**
Casper Hansen **Christian Hansen** **Jakob Grue Simonsen**
Department of Computer Science
University of Copenhagen
{augenstein,c.lioma,wang,lcl,c.hansen,chrh,simonsen}@di.ku.dk

Abstract

We contribute the largest publicly available dataset of naturally occurring factual claims for the purpose of automatic claim verification. It is collected from 26 fact checking websites in English, paired with textual sources and rich metadata, and labelled for veracity by human expert journalists. We present an in-depth analysis of the dataset, highlighting characteristics and challenges. Further, we present results for automatic veracity prediction, both with established baselines and with a novel method for joint ranking of evidence pages and predicting veracity that outperforms all baselines. Significant performance increases are achieved by encoding evidence, and by modelling metadata. Our best-performing model achieves a Macro F1 of 49.2%, showing that this is a challenging testbed for claim veracity prediction.

1 Introduction

Misinformation and disinformation are two of the most pertinent and difficult challenges of the information age, exacerbated by the popularity of social media. In an effort to counter this, a significant amount of manual labour has been invested in fact checking claims, often collecting the results of these manual checks on fact checking portals or websites such as politifact.com or snopes.com. In a parallel development, researchers have recently started to view fact checking as a task that can be partially automated, using machine learning and NLP to automatically predict the *veracity* of claims. However, existing efforts either use small datasets consisting of naturally occurring claims (e.g. Mihalcea and Strapparava (2009); Zubiaga et al. (2016)), or datasets consisting of artificially constructed claims such as FEVER (Thorne et al., 2018). While the latter offer valuable contributions to further automatic claim verification work, they cannot replace real-world datasets.

Feature	Value
ClaimID	farg-00004
Claim	Mexico and Canada assemble cars with foreign parts and send them to the U.S. with no tax.
Label	distorts
Claim URL	https://www.factcheck.org/2018/10/factchecking-trump-on-trade/
Reason	None
Category	the-factcheck-wire
Speaker	Donald Trump
Checker	Eugene Kiely
Tags	North American Free Trade Agreement
Claim Entities	United_States, Canada, Mexico
Article Title	FactChecking Trump on Trade
Publish Date	October 3, 2018
Claim Date	Monday, October 1, 2018

Table 1: An example of a claim instance. Entities are obtained via entity linking. Article and outlink texts, evidence search snippets and pages are not shown.

Contributions. We introduce the currently largest claim verification dataset of naturally occurring claims.[1] It consists of 34,918 claims, collected from 26 fact checking websites in English; evidence pages to verify the claims; the context in which they occurred; and rich metadata (see Table 1 for an example). We perform a thorough analysis to identify characteristics of the dataset such as entities mentioned in claims. We demonstrate the utility of the dataset by training state of the art veracity prediction models, and find that evidence pages as well as metadata significantly contribute to model performance. Finally, we propose a novel model that jointly ranks evidence pages and performs veracity prediction. The best-performing model achieves a Macro F1 of 49.2%, showing that this is a non-trivial dataset with remaining challenges for future work.

[1] The dataset is found here: https://copenlu.github.io/publication/2019_emnlp_augenstein/

2 Related Work

2.1 Datasets

Over the past few years, a variety of mostly small datasets related to fact checking have been released. An overview over core datasets is given in Table 2. The datasets can be grouped into four categories (I–IV). Category I contains datasets aimed at testing how well the veracity[3] of a claim can be predicted using the claim alone, without context or evidence documents. Category II contains datasets bundled with documents related to each claim – either topically related to provide context, or serving as evidence. Those documents are, however, not annotated. Category III is for predicting veracity; they encourage retrieving evidence documents as part of their task description, but do not distribute them. Finally, category IV comprises datasets annotated for both veracity and stance. Thus, every document is annotated with a label indicating whether the document supports or denies the claim, or is unrelated to it. Additional labels can then be added to the datasets to better predict veracity, for instance by jointly training stance and veracity prediction models.

Methods not shown in the table, but related to fact checking, are stance detection for claims (Ferreira and Vlachos, 2016; Pomerleau and Rao, 2017; Augenstein et al., 2016a; Kochkina et al., 2017; Augenstein et al., 2016b; Zubiaga et al., 2018; Riedel et al., 2017), satire detection (Rubin et al., 2016), clickbait detection (Karadzhov et al., 2017), conspiracy news detection (Tacchini et al., 2017), rumour cascade detection (Vosoughi et al., 2018) and claim perspectives detection (Chen et al., 2019).

Claims are obtained from a variety of sources, including Wikipedia, Twitter, criminal reports and fact checking websites such as politifact.com and snopes.com. The same goes for documents – these are often websites obtained through Web search queries, or Wikipedia documents, tweets or Facebook posts. Most datasets contain a fairly small number of claims, and those that do not, often lack evidence documents. An exception is Thorne et al. (2018), who create a Wikipedia-based fact checking dataset. While a good testbed for developing deep neural architectures, their dataset is artificially constructed and can thus not take metadata about claims into account.

Contributions: We provide a dataset that, uniquely among extant datasets, contains a large number of *naturally occurring* claims and rich additional meta-information.

2.2 Methods

Fact checking methods partly depend on the type of dataset used. Methods only taking into account claims typically encode those with CNNs or RNNs (Wang, 2017; Pérez-Rosas et al., 2018), and potentially encode metadata (Wang, 2017) in a similar way. Methods for small datasets often use handcrafted features that are a mix of bag of word and other lexical features, e.g. LIWC, and then use those as input to a SVM or MLP (Mihalcea and Strapparava, 2009; Pérez-Rosas et al., 2018; Baly et al., 2018). Some use additional Twitter-specific features (Enayet and El-Beltagy, 2017). More involved methods taking into account evidence documents, often trained on larger datasets, consist of evidence identification and ranking following a neural model that measures the compatibility between claim and evidence (Thorne et al., 2018; Mihaylova et al., 2018; Yin and Roth, 2018).

Contributions: The latter category above is the most related to our paper as we consider evidence documents. However, existing models are not trained jointly for evidence identification, or for stance and veracity prediction, but rather employ a pipeline approach. Here, we show that a joint approach that learns to weigh evidence pages by their importance for veracity prediction can improve downstream veracity prediction performance.

3 Dataset Construction

We crawled a total of 43,837 claims with their metadata (see details in Table 11). We present the data collection in terms of selecting sources, crawling claims and associated metadata (Section 3.1); retrieving evidence pages; and linking entities in the crawled claims (Section 3.3).

3.1 Selection of sources

We crawled all active fact checking websites in English listed by Duke Reporters' Lab[4] and on the Fact Checking Wikipedia page.[5] This resulted in

[3] We use *veracity*, *claim credibility*, and *fake news* prediction interchangeably here – these terms are often conflated in the literature and meant to have the same meaning.

[4] https://reporterslab.org/fact-checking/

[5] https://en.wikipedia.org/wiki/Fact_checking

Dataset	# Claims	Labels	metadata	Claim Sources
I: Veracity prediction w/o evidence				
Wang (2017)	12,836	6	Yes	Politifact
Pérez-Rosas et al. (2018)	980	2	No	News Websites
II: Veracity				
Bachenko et al. (2008)	275	2	No	Criminal Reports
Mihalcea and Strapparava (2009)	600	2	No	Crowd Authors
Mitra and Gilbert (2015)†	1,049	5	No	Twitter
Ciampaglia et al. (2015)†	10,000	2	No	Google, Wikipedia
Popat et al. (2016)	5,013	2	Yes	Wikipedia, Snopes
Shu et al. (2018)†	23,921	2	Yes	Politifact, gossipcop.com
Datacommons Fact Check[2]	10,564	2-6	Yes	Fact Checking Websites
III: Veracity (evidence encouraged, but not provided)				
Barrn-Cedeo et al. (2018)	150	3	No	factcheck.org, Snopes
IV: Veracity + stance				
Vlachos and Riedel (2014)	106	5	Yes	Politifact, Channel 4 News
Zubiaga et al. (2016)	330	3	Yes	Twitter
Derczynski et al. (2017)	325	3	Yes	Twitter
Baly et al. (2018)	422	2	No	ara.reuters.com, verify-sy.com
Thorne et al. (2018)†	185,445	3	No	Wikipedia
V: Veracity + evidence relevancy				
MultiFC	36,534	2-40	Yes	Fact Checking Websites

Table 2: Comparison of fact checking datasets. † indicates claims are not 'naturally occuring': Mitra and Gilbert (2015) use events as claims; Ciampaglia et al. (2015) use DBPedia tiples as claims; Shu et al. (2018) use tweets as claims; and Thorne et al. (2018) rewrite sentences in Wikipedia as claims.

38 websites in total (shown in Table 11). Out of these, ten websites could not be crawled, as further detailed in Table 9. In the later experimental descriptions, we refer to the part of the dataset crawled from a specific fact checking website as a *domain*, and we refer to each website as *source*.

From each source, we crawled the ID, claim, label, URL, reason for label, categories, person making the claim (speaker), person fact checking the claim (checker), tags, article title, publication date, claim date, as well as the full text that appears when the claim is clicked. Lastly, the above full text contains hyperlinks, so we further crawled the full text that appears when each of those hyperlinks are clicked (outlinks).

There were a number of crawling issues, e.g. security protection of websites with SSL/TLS protocols, time out, URLs that pointed to pdf files instead of HTML content, or unresolvable encoding. In all of these cases, the content could not be retrieved. For some websites, no veracity labels were available, in which case, they were not selected as domains for training a veracity prediction model. Moreover, not all types of metadata (category, speaker, checker, tags, claim date, publish date) were available for all websites; and availability of articles and full texts differs as well.

We performed semi-automatic cleansing of the dataset as follows. First, we double-checked that the veracity labels would not appear in claims. For some domains, the first or last sentence of the claim would sometimes contain the veracity label, in which case we would discard either the full sentence or part of the sentence. Next, we checked the dataset for duplicate claims. We found 202 such instances, 69 of them with different labels. Upon manual inspection, this was mainly due to them appearing on different websites, with labels not differing much in practice (e.g. 'Not true', vs. 'Mostly False'). We made sure that all such duplicate claims would be in the training split of the dataset, so that the models would not have an unfair advantage. Finally, we performed some minor manual merging of label types for the same domain where it was clear that they were supposed to denote the same level of veracity (e.g. 'distorts', 'distorts the facts').

This resulted in a total of 36,534 claims with their metadata. For the purposes of fact verification, we discarded instances with labels that occur fewer than 5 times, resulting in 34,918 claims. The number of instances, as well as labels per domain, are shown in Table 6 and label names in Table 10 in the appendix. The dataset is split into a training part (80%) and a development and testing part (10% each) in a label-stratified manner. Note that the domains vary in the number of labels, ranging from 2 to 27. Labels include both straight-forward ratings of veracity (‘correct’, ‘incorrect’), but also labels that would be more difficult to map onto a veracity scale (e.g. ‘grass roots movement!’, ‘misattributed’, ‘not the whole story’). We therefore do not postprocess label types across domains to map them onto the same scale, and rather treat them as is. In the methodology section (Section 4), we show how a model can be trained on this dataset regardless by framing this multi-domain veracity prediction task as a multi-task learning (MTL) one.

3.2 Retrieving Evidence Pages

The text of each claim is submitted verbatim as a query to the Google Search API (without quotes). The 10 most highly ranked search results are retrieved, for each of which we save the title; Google search rank; URL; time stamp of last update; search snippet; as well as the full Web page. We acknowledge that search results change over time, which might have an effect on veracity prediction. However, studying such temporal effects is outside the scope of this paper. Similar to Web crawling claims, as described in Section 3.1, the corresponding Web pages can in some cases not be retrieved, in which case fewer than 10 evidence pages are available. The resulting evidence pages are from a wide variety of URL domains, though with a predictable skew towards popular websites, such as Wikipedia or The Guardian (see Table 3 for detailed statistics).

3.3 Entity Detection and Linking

To better understand what claims are about, we conduct entity linking for all claims. Specifically, mentions of people, places, organisations, and other named entities within a claim are recognised and linked to their respective Wikipedia pages, if available. Where there are different entities with the same name, they are disambiguated. For this, we apply the state-of-the-art neural entity linking model by Kolitsas et al. (2018). This results in a total of 25,763 entities detected and linked to Wikipedia, with a total of 15,351 claims involved, meaning that 42% of all claims contain entities that can be linked to Wikipedia. Later on, we use entities as additional metadata (see Section 4.3). The distribution of claim numbers according to the number of entities they contain is shown in Figure 1. We observe that the majority of claims have one to four entities, and the maximum number of 35 entities occurs in one claim only. Out of the 25,763 entities, 2,767 are unique entities. The top 30 most frequent entities are listed in Table 4. This clearly shows that most of the claims involve entities related to the United States, which is to be expected, as most of the fact checking websites are US-based.

Domain	%
https://en.wikipedia.org/	4.425
https://www.snopes.com/	3.992
https://www.washingtonpost.com/	3.025
https://www.nytimes.com/	2.478
https://www.theguardian.com/	1.807
https://www.youtube.com/	1.712
https://www.dailymail.co.uk/	1.558
https://www.usatoday.com/	1.279
https://www.politico.com/	1.241
http://www.politifact.com/	1.231
https://www.pinterest.com/	1.169
https://www.factcheck.org/	1.09
https://www.gossipcop.com/	1.073
https://www.cnn.com/	1.065
https://www.npr.org/	0.957
https://www.forbes.com/	0.911
https://www.vox.com/	0.89
https://www.theatlantic.com/	0.88
https://twitter.com/	0.767
https://www.hoax-slayer.net/	0.655
http://time.com/	0.554
https://www.bbc.com/	0.551
https://www.nbcnews.com/	0.515
https://www.cnbc.com/	0.514
https://www.cbsnews.com/	0.503
https://www.facebook.com/	0.5
https://www.newyorker.com/	0.495
https://www.foxnews.com/	0.468
https://people.com/	0.439
http://www.cnn.com/	0.419

Table 3: The top 30 most frequently occurring URL domains.

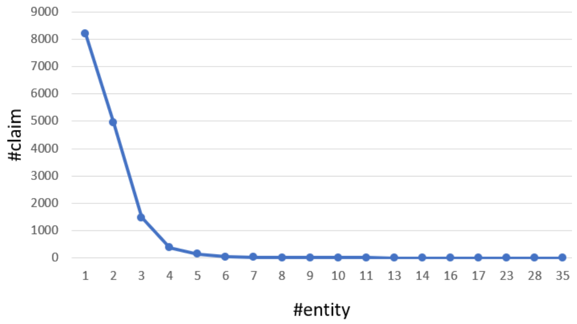


Figure 1: Distribution of entities in claims.

4 Claim Veracity Prediction

We train several models to predict the veracity of claims. Those fall into two categories: those that

Entity	Frequency
United_States	2810
Barack_Obama	1598
Republican_Party_(United_States)	783
Texas	665
Democratic_Party_(United_States)	560
Donald_Trump	556
Wisconsin	471
United_States_Congress	354
Hillary_Rodham_Clinton	306
Bill_Clinton	292
California	285
Russia	275
Ohio	239
China	229
George_W._Bush	208
Medicare_(United_States)	206
Australia	186
Iran	183
Brad_Pitt	180
Islam	178
Iraq	176
Canada	174
White_House	166
New_York_City	164
Washington,_D.C.	164
Jennifer_Aniston	163
Mexico	158
Ted_Cruz	152
Federal_Bureau_of_Investigation	146
Syria	130

Table 4: Top 30 most frequent entities listed by their Wikipedia URL with prefix omitted

only consider the claims themselves, and those that encode evidence pages as well. In addition, claim metadata (speaker, checker, linked entities) is optionally encoded for both categories of models, and ablation studies with and without that metadata are shown. We first describe the base model used in Section 4.1, followed by introducing our novel evidence ranking and veracity prediction model in Section 4.2, and lastly the metadata encoding model in Section 4.3.

4.1 Multi-Domain Claim Veracity Prediction with Disparate Label Spaces

Since not all fact checking websites use the same claim labels (see Table 6, and Table 10 in the appendix), training a claim veracity prediction model is not entirely straight-forward. One option would be to manually map those labels onto one another. However, since the sheer number of labels is rather large (165), and it is not always clear from the guidelines on fact checking websites how they can be mapped onto one another, we opt to learn how these labels relate to one another as part of the veracity prediction model. To do so, we employ the multi-task learning (MTL) approach inspired by collaborative filtering presented in Augenstein et al. (2018) (*MTL with LEL*–multitask learning with label embedding layer) that excels on pairwise sequence classification tasks with disparate label spaces. More concretely, each domain is modelled as its own task in a MTL architecture, and labels are projected into a fixed-length label embedding space. Predictions are then made by taking the dot product between the claim-evidence embeddings and the label embeddings. By doing so, the model implicitly learns how semantically close the labels are to one another, and can benefit from this knowledge when making predictions for individual tasks, which on their own might only have a small number of instances. When making predictions for individual domains/tasks, both at training and at test time, as well as when calculating the loss, a mask is applied such that the valid and invalid labels for that task are restricted to the set of known task labels.

Note that the setting here slightly differs from Augenstein et al. (2018). There, tasks are less strongly related to one another; for example, they consider stance detection, aspect-based sentiment analysis and natural language inference. Here, we have different domains, as opposed to conceptually different tasks, but use their framework, as we have the same underlying problem of disparate label spaces. A more formal problem definition follows next, as our evidence ranking and veracity prediction model in Section 4.2 then builds on it.

4.1.1 Problem Definition

We frame our problem as a multi-task learning one, where access to labelled datasets for T tasks $\mathcal{T}_1, \ldots, \mathcal{T}_T$ is given at training time with a target task $\mathcal{T}_T$ that is of particular interest. The training dataset for task $\mathcal{T}_i$ consists of N examples $X_{\mathcal{T}_i} = \{x_1^{\mathcal{T}_i}, \ldots, x_N^{\mathcal{T}_i}\}$ and their labels $Y_{\mathcal{T}_i} = \{\mathbf{y}_1^{\mathcal{T}_i}, \ldots, \mathbf{y}_N^{\mathcal{T}_i}\}$. The base model is a classic deep neural network MTL model (Caruana, 1993) that shares its parameters across tasks and has task-specific softmax output layers that output a probability distribution $\mathbf{p}^{\mathcal{T}_i}$ for task $\mathcal{T}_i$:

$$\mathbf{p}^{\mathcal{T}_i} = \text{softmax}(\mathbf{W}^{\mathcal{T}_i}\mathbf{h} + \mathbf{b}^{\mathcal{T}_i}) \qquad (1)$$

where $\text{softmax}(\mathbf{x}) = e^{\mathbf{x}} / \sum_{i=1}^{\|\mathbf{x}\|} e^{\mathbf{x}_i}$, $\mathbf{W}^{\mathcal{T}_i} \in \mathbb{R}^{L_i \times h}$, $\mathbf{b}^{\mathcal{T}_i} \in \mathbb{R}^{L_i}$ is the weight matrix and bias term of the output layer of task $\mathcal{T}_i$ respectively, $\mathbf{h} \in \mathbb{R}^h$ is the jointly learned hidden representation, L_i is the number of labels for task $\mathcal{T}_i$, and h is the dimensionality of $\mathbf{h}$. The MTL model is trained to minimise the sum of individual task losses $\mathcal{L}_1 + \ldots + \mathcal{L}_T$ using a negative log-likelihood objective.

Label Embedding Layer. To learn the relationships between labels, a Label Embedding Layer (LEL) embeds labels of all tasks in a joint Euclidian space. Instead of training separate softmax output layers as above, a label compatibility function $c(\cdot, \cdot)$ measures how similar a label with embedding $\mathbf{l}$ is to the hidden representation $\mathbf{h}$:

$$c(\mathbf{l}, \mathbf{h}) = \mathbf{l} \cdot \mathbf{h} \tag{2}$$

where $\cdot$ is the dot product. Padding is applied such that l and h have the same dimensionality. Matrix multiplication and softmax are used for making predictions:

$$\mathbf{p} = \mathrm{softmax}(\mathbf{L}\mathbf{h}) \tag{3}$$

where $\mathbf{L} \in \mathbb{R}^{(\sum_i L_i) \times l}$ is the label embedding matrix for all tasks and l is the dimensionality of the label embeddings. We apply a task-specific mask to $\mathbf{L}$ in order to obtain a task-specific probability distribution $\mathbf{p}^{\mathcal{T}_i}$. The LEL is shared across all tasks, which allows the model to learn the relationships between labels in the joint embedding space.

4.2 Joint Evidence Ranking and Claim Veracity Prediction

So far, we have ignored the issue of how to obtain claim representation, as the base model described in the previous section is agnostic to how instances are encoded. A very simple approach, which we report as a baseline, is to encode claim texts only. Such a model ignores evidence for and against a claim, and ends up guessing the veracity based on surface patterns observed in the claim texts.

We next introduce two variants of evidence-based veracity prediction models that encode 10 pieces of evidence in addition to the claim. Here, we opt to encode search snippets as opposed to whole retrieved pages. While the latter would also be possible, it comes with a number of additional challenges, such as encoding large documents, parsing tables or PDF files, and encoding images or videos on these pages, which we leave to future work. Search snippets also have the benefit that they already contain summaries of the part of the page content that is most related to the claim.

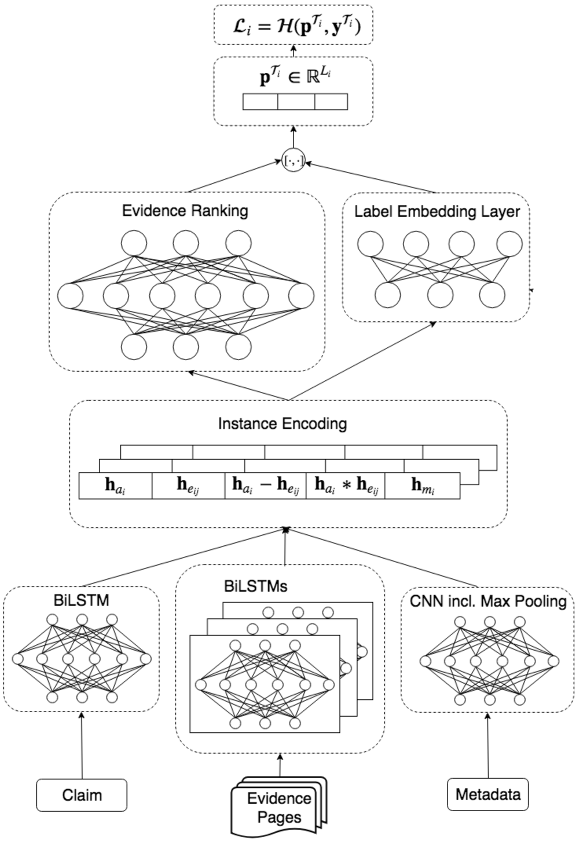

$\mathcal{L}_i = \mathcal{H}(\mathbf{p}^{\mathcal{T}_i}, \mathbf{y}^{\mathcal{T}_i})$
$\mathbf{p}^{\mathcal{T}_i} \in \mathbb{R}^{L_i}$
$[\cdot,\cdot]$
Evidence Ranking
Label Embedding Layer
Instance Encoding
$\mathbf{h}_{a_i}$
$\mathbf{h}_{e_{ij}}$
$\mathbf{h}_{a_i} - \mathbf{h}_{e_{ij}}$
$\mathbf{h}_{a_i} * \mathbf{h}_{e_{ij}}$
$\mathbf{h}_{m_i}$
BiLSTM
BiLSTMs
CNN incl. Max Pooling
Claim
Evidence Pages
Metadata

Figure 2: The Joint Veracity Prediction and Evidence Ranking model, shown for one task.

4.2.1 Problem Definition

Our problem is to obtain encodings for N examples $X_{\mathcal{T}_i} = \{x_1^{\mathcal{T}_i}, \ldots, x_N^{\mathcal{T}_i}\}$. For simplicity, we will henceforth drop the task superscript and refer to instances as $X = \{x_1, \ldots, x_N\}$, as instance encodings are learned in a task-agnostic fashion. Each example further consists of a claim a_i and $k = 10$ evidence pages $E_k = \{e_{1_0}, \ldots, e_{N_{10}}\}$.

Each claim and evidence page is encoded with a BiLSTM to obtain a sentence embedding, which is the concatenation of the last state of the forward and backward reading of the sentence, i.e. $\mathbf{h} = BiLSTM(\cdot)$, where $\mathbf{h}$ is the sentence embedding.

Next, we want to combine claims and evidence sentence embeddings into joint instance representations. In the simplest case, referred to as model variant *crawled_avg*, we mean average the BiLSTM sentence embeddings of all evidence pages (signified by the overline) and concatenate those with the claim embeddings, i.e.

$$\mathbf{s}_{g_i} = [\mathbf{h}_{a_i}; \overline{\mathbf{h}_{E_i}}] \tag{4}$$

where s_{g_i} is the resulting encoding for training example i and $[\cdot; \cdot]$ denotes vector concatenation.

However, this has the disadvantage that all evidence pages are considered equal.

Evidence Ranking The here proposed alternative instance encoding model, *crawled_ranked*, which achieves the highest overall performance as discussed in Section 5, learns the compatibility between an instance's claim and each evidence page. It ranks evidence pages by their utility for the veracity prediction task, and then uses the resulting ranking to obtain a weighted combination of all claim-evidence pairs. No direct labels are available to learn the ranking of individual documents, only for the veracity of the associated claim, so the model has to learn evidence ranks implicitly.

To combine claim and evidence representations, we use the matching model proposed for the task of natural language inference by Mou et al. (2016) and adapt it to combine an instance's claim representation with each evidence representation, i.e.

$$s_{r_{i_j}} = [\mathbf{h}_{a_i}; \mathbf{h}_{e_{i_j}}; \mathbf{h}_{a_i} - \mathbf{h}_{e_{i_j}}; \mathbf{h}_{a_i} \cdot \mathbf{h}_{e_{i_j}}] \qquad (5)$$

where $s_{r_{i_j}}$ is the resulting encoding for training example i and evidence page j , $[\cdot;\cdot]$ denotes vector concatenation, and $\cdot$ denotes the dot product.

All joint claim-evidence representations $\mathbf{s}_{r_{i_0}}, \ldots, \mathbf{s}_{r_{i_{10}}}$ are then projected into the binary space via a fully connected layer FC, followed by a non-linear activation function f, to obtain a soft ranking of claim-evidence pairs, in practice a 10-dimensional vector,

$$\mathbf{o}_i = [f(\mathrm{FC}(s_{r_{i_0}})); \ldots; f(\mathrm{FC}(s_{r_{i_{10}}}))] \qquad (6)$$

where $[\cdot;\cdot]$ denotes concatenation.

Scores for all labels are obtained as per (6) above, with the same input instance embeddings as for the evidence ranker, i.e. $s_{r_{i_j}}$. Final predictions for all claim-evidence pairs are then obtained by taking the dot product between the label scores and binary evidence ranking scores, i.e.

$$\mathbf{p}_i = \mathrm{softmax}(c(\mathbf{l}, \mathbf{s}_{\mathbf{r_i}}) \cdot \mathbf{o}_i) \qquad (7)$$

Note that the novelty here is that, unlike for the model described in Mou et al. (2016), we have no direct labels for learning weights for this matching model. Rather, our model has to implicitly learn these weights for each claim-evidence pair in an end-to-end fashion given the veracity labels.

Model	Micro F1	Macro F1
claim-only	0.469	0.253
claim-only_embavg	0.384	0.302
crawled-docavg	0.438	0.248
crawled_ranked	0.613	0.441
claim-only + meta	0.494	0.324
claim-only_embavg + meta	0.418	0.333
crawled-docavg + meta	0.483	0.286
crawled_ranked + meta	**0.625**	**0.492**

Table 5: Results with different model variants on the test set, 'meta' means all metadata is used.

4.3 Metadata

We experiment with how useful claim metadata is, and encode the following as one-hot vectors: speaker, category, tags and linked entities. We do not encode 'Reason' as it gives away the label, and do not include 'Checker' as there are too many unique checkers for this information to be relevant. The claim publication date is potentially relevant, but it does not make sense to merely model this as a one-hot feature, so we leave incorporating temporal information to future work. Since all metadata consists of individual words and phrases, a sequence encoder is not necessary, and we opt for a CNN followed by a max pooling operation as used in Wang (2017) to encode metadata for fact checking. The max-pooled metadata representations, denoted h_m, are then concatenated with the instance representations, e.g. for the most elaborate model, *crawled_ranked*, these would be concatenated with $s_{cr_{i_j}}$.

5 Experiments

5.1 Experimental Setup

The base sentence embedding model is a BiLSTM over all words in the respective sequences with randomly initialised word embeddings, following Augenstein et al. (2018). We opt for this strong baseline sentence encoding model, as opposed to engineering sentence embeddings that work particularly well for this dataset, to showcase the dataset. We would expect pre-trained contextual encoding models, e.g. ELMO (Peters et al., 2018), ULMFit (Howard and Ruder, 2018), BERT (Devlin et al., 2018), to offer complementary performance gains, as has been shown for a few recent papers (Wang et al., 2018a; Rajpurkar et al., 2018).

For claim veracity prediction without evidence documents with the MTL with LEL model, we use the following sentence encoding variants: *claim-*

only, which uses a BiLSTM-based sentence embedding as input, and *claim-only_embavg*, which uses a sentence embedding based on mean averaged word embeddings as input.

We train one multi-task model per task (i.e., one model per domain). We perform a grid search over the following hyperparameters, tuned on the respective dev set, and evaluate on the correspoding test set (final settings are underlined): word embedding size [64, 128, 256], BiLSTM hidden layer size [64, 128, 256], number of BiLSTM hidden layers [1, 2, 3], BiLSTM dropout on input and output layers [0.0, 0.1, 0.2, 0.5], word-by-word-attention for BiLSTM with window size 10 (Bahdanau et al., 2014) [True, False], skip-connections for the BiLSTM [True, False], batch size [32, 64, 128], label embedding size [16, 32, 64]. We use ReLU as an activation function for both the BiLSTM and the CNN. For the CNN, the following hyperparameters are used: number filters [32], kernel size [32]. We train using cross-entropy loss and the RMSProp optimiser with initial learning rate of 0.001 and perform early stopping on the dev set with a patience of 3.

5.2 Results

For each domain, we compute the Micro as well as Macro F1, then mean average results over all domains. Core results with all vs. no metadata are shown in Table 5. We first experiment with different base model variants and find that label embeddings improve results, and that the best proposed models utilising multiple domains outperform single-task models (see Table 8). This corroborates the findings of Augenstein et al. (2018). Per-domain results with the best model are shown in Table 6. Domain names are from hereon after abbreviated for brevity, see Table 11 in the appendix for correspondences to full website names. Unsurprisingly, it is hard to achieve a high Macro F1 for domains with many labels, e.g. tron and snes. Further, some domains, surprisingly mostly with small numbers of instances, seem to be very easy – a perfect Micro and Macro F1 score of 1.0 is achieved on ranz, bove, buca, fani and thal. We find that for those domains, the verdict is often already revealed as part of the claim using explicit wording.

Domain	# Insts	# Labs	Micro F1	Macro F1
ranz	21	2	1.000	1.000
bove	295	2	1.000	1.000
abbc	436	3	0.463	0.453
huca	34	3	1.000	1.000
mpws	47	3	0.667	0.583
peck	65	3	0.667	0.472
faan	111	3	0.682	0.679
clck	38	3	0.833	0.619
fani	20	3	1.000	1.000
chct	355	4	0.550	0.513
obry	59	4	0.417	0.268
vees	504	4	0.721	0.425
faly	111	5	0.278	0.5
goop	2943	6	0.822	0.387
pose	1361	6	0.438	0.328
thet	79	6	0.55	0.37
thal	163	7	1.000	1.000
afck	433	7	0.357	0.259
hoer	1310	7	0.694	0.549
para	222	7	0.375	0.311
wast	201	7	0.344	0.214
vogo	654	8	0.594	0.297
pomt	15390	9	0.321	0.276
snes	6455	12	0.551	0.097
farg	485	11	0.500	0.140
tron	3423	27	0.429	0.046
avg		7.17	0.625	0.492

Table 6: Total number of instances and unique labels per domain, as well as per-domain results with model *crawled_ranked* + *meta*, sorted by label size

Metadata	Micro F1	Macro F1
None	**0.627**	0.441
Speaker	0.602	0.435
+ Tags	0.608	0.460
Tags	0.585	0.461
Entity	0.569	0.427
+ Speaker	0.607	0.477
+ Tags	0.625	**0.492**

Table 7: Ablation results with base model *crawled_ranked* for different types of metadata

Model	Micro F1	Macro F1
STL	0.527	0.388
MTL	0.556	0.448
MTL + LEL	**0.625**	**0.492**

Table 8: Ablation results with *crawled_ranked* + *meta* encoding for STL vs. MTL vs. MTL + LEL training

Claim-Only vs. Evidence-Based Veracity Prediction. Our evidence-based claim veracity prediction models outperform claim-only veracity prediction models by a large margin. Unsurprisingly, *claim-only_embavg* is outperformed by *claim-only*. Further, *crawled_ranked* is our best-performing model in terms of Micro F1 and Macro F1, meaning that our model captures that not every piece of evidence is equally important, and can

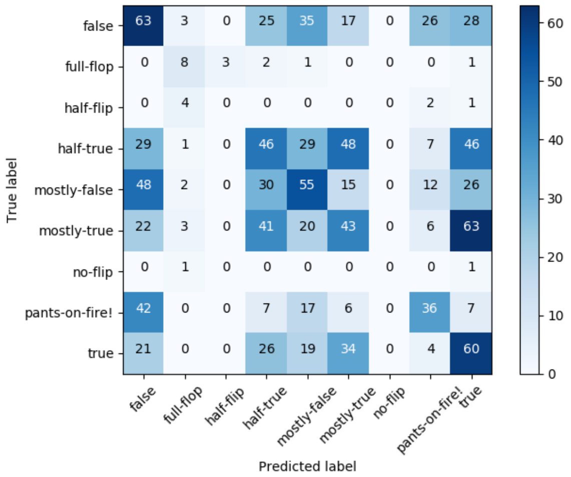


Figure 3: Confusion matrix of predicted labels with best-performing model, *crawled_ranked* + *meta*, on the 'pomt' domain

utilise this for veracity prediction.

Metadata. We perform an ablation analysis of how metadata impacts results, shown in Table 7. Out of the different types of metadata, topic tags on their own contribute the most. This is likely because they offer highly complementary information to the claim text of evidence pages. Only using all metadata together achieves a higher Macro F1 at similar Micro F1 than using no metadata at all. To further investigate this, we split the test set into those instances for which no metadata is available vs. those for which metadata is available. We find that encoding metadata within the model hurts performance for domains where no metadata is available, but improves performance where it is. In practice, an ensemble of both types of models would be sensible, as well as exploring more involved methods of encoding metadata.

6 Analysis and Discussion

An analysis of labels frequently confused with one another, for the largest domain 'pomt' and best-performing model *crawled_ranked* + *meta* is shown in Figure 3. The diagonal represents when gold and predicted labels match, and the numbers signify the number of test instances. One can observe that the model struggles more to detect claims with labels 'true' than those with label 'false'. Generally, many confusions occur over close labels, e.g. 'half-true' vs. 'mostly true'.

We further analyse what properties instances that are predicted correctly vs. incorrectly have, using the model *crawled_ranked meta*. We find that, unsurprisingly, longer claims are harder to classify correctly, and that claims with a high direct token overlap with evidence pages lead to a high evidence ranking. When it comes to frequently occurring tags and entities, very general tags such as 'government-and-politics' or 'tax' that do not give away much, frequently co-occur with incorrect predictions, whereas more specific tags such as 'brisbane-4000' or 'hong-kong' tend to co-occur with correct predictions. Similar trends are observed for bigrams. This means that the model has an easy time succeeding for instances where the claims are short, where specific topics tend to co-occur with certain veracities, and where evidence documents are highly informative. Instances with longer, more complex claims where evidence is ambiguous remain challenging.

7 Conclusions

We present a new, real-world fact checking dataset, currently the largest of its kind. It consists of 34,918 claims collected from 26 fact checking websites, rich metadata and 10 retrieved evidence pages per claim. We find that encoding the metadata as well evidence pages helps, and introduce a new joint model for ranking evidence pages and predicting veracity.

Acknowledgments

This research is partially supported by QUARTZ (721321, EU H2020 MSCA-ITN) and DABAI (5153-00004A, Innovation Fund Denmark).

References

Isabelle Augenstein, Tim Rocktäschel, Andreas Vlachos, and Kalina Bontcheva. 2016a. Stance Detection with Bidirectional Conditional Encoding. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 876–885, Austin, Texas. Association for Computational Linguistics.

Isabelle Augenstein, Sebastian Ruder, and Anders Søgaard. 2018. Multi-Task Learning of Pairwise Sequence Classification Tasks over Disparate Label Spaces. In *NAACL-HLT*, pages 1896–1906. Association for Computational Linguistics.

Isabelle Augenstein, Andreas Vlachos, and Kalina Bontcheva. 2016b. USFD at SemEval-2016 Task 6: Any-Target Stance Detection on Twitter with Autoencoders. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 389–393, San Diego, California. Association for Computational Linguistics.

Joan Bachenko, Eileen Fitzpatrick, and Michael Schonwetter. 2008. Verification and implementation of language-based deception indicators in civil and criminal narratives. In *Proceedings of the 22nd International Conference on Computational Linguistics-Volume 1*, pages 41–48. Association for Computational Linguistics.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural Machine Translation by Jointly Learning to Align and Translate. In *Proceedings of ICLR*.

Ramy Baly, Mitra Mohtarami, James R. Glass, Lluís Màrquez, Alessandro Moschitti, and Preslav Nakov. 2018. Integrating Stance Detection and Fact Checking in a Unified Corpus. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 2 (Short Papers)*, pages 21–27. Association for Computational Linguistics.

Alberto Barrn-Cedeo, Tamer Elsayed, Reem Suwaileh, Llus Mrquez, Pepa Atanasova, Wajdi Zaghouani, Spas Kyuchukov, Giovanni Da San Martino, and Preslav Nakov. 2018. Overview of the CLEF-2018 CheckThat! Lab on automatic identification and verification of political claims. Task 2: Factuality. In *CLEF (Working Notes)*, volume 2125 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Rich Caruana. 1993. Multitask Learning: A Knowledge-Based Source of Inductive Bias. In *Proceedings of ICML*.

Sihao Chen, Daniel Khashabi, Wenpeng Yin, Chris Callison-Burch, and Dan Roth. 2019. Seeing Things from a Different Angle: Discovering Diverse Perspectives about Claims. In *Proceedings of NAACL*.

G L Ciampaglia, P Shiralkar, L M Rocha, J Bollen, F Menczer, and A Flammini. 2015. Computational Fact Checking from Knowledge Networks. *PLoS One*, 10(6).

Leon Derczynski, Kalina Bontcheva, Maria Liakata, Rob Procter, Geraldine Wong Sak Hoi, and Arkaitz Zubiaga. 2017. SemEval-2017 Task 8: RumourEval: Determining rumour veracity and support for rumours. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 69–76. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *CoRR*, abs/1810.04805.

Omar Enayet and Samhaa R. El-Beltagy. 2017. NileTMRG at SemEval-2017 Task 8: Determining Rumour and Veracity Support for Rumours on Twitter. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 470–474. Association for Computational Linguistics.

William Ferreira and Andreas Vlachos. 2016. Emergent: a novel data-set for stance classification. In *HLT-NAACL*, pages 1163–1168. The Association for Computational Linguistics.

Andreas Hanselowski, PVS Avinesh, Benjamin Schiller, and Felix Caspelherr. 2017. Team Athene on the Fake News Challenge.

Jeremy Howard and Sebastian Ruder. 2018. Universal Language Model Fine-tuning for Text Classification. In *ACL (1)*, pages 328–339. Association for Computational Linguistics.

Georgi Karadzhov, Pepa Gencheva, Preslav Nakov, and Ivan Koychev. 2017. We Built a Fake News / Click Bait Filter: What Happened Next Will Blow Your Mind! In *RANLP 2017*, pages 334–343.

Hamid Karimi, Proteek Roy, Sari Saba-Sadiya, and Jiliang Tang. 2018. Multi-Source Multi-Class Fake News Detection. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1546–1557.

Elena Kochkina, Maria Liakata, and Isabelle Augenstein. 2017. Turing at SemEval-2017 Task 8: Sequential Approach to Rumour Stance Classification with Branch-LSTM. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 475–480, Vancouver, Canada. Association for Computational Linguistics.

Nikolaos Kolitsas, Octavian-Eugen Ganea, and Thomas Hofmann. 2018. End-to-End Neural Entity Linking. In *Proceedings of the 22nd Conference on Computational Natural Language Learning*, pages 519–529. Association for Computational Linguistics.

Rada Mihalcea and Carlo Strapparava. 2009. The lie detector: Explorations in the automatic recognition of deceptive language. In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, pages 309–312. Association for Computational Linguistics.

Tsvetomila Mihaylova, Preslav Nakov, Lluís Màrquez, Alberto Barrón-Cedeño, Mitra Mohtarami, Georgi Karadzhov, and James R. Glass. 2018. Fact Checking in Community Forums. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, New Orleans, Louisiana, USA, February 2-7, 2018.* AAAI Press.

Tanushree Mitra and Eric Gilbert. 2015. Credbank: A large-scale social media corpus with associated credibility annotations. In *ICWSM*, pages 258–267.

Lili Mou, Rui Men, Ge Li, Yan Xu, Lu Zhang, Rui Yan, and Zhi Jin. 2016. Natural Language Inference by Tree-Based Convolution and Heuristic Matching. In *ACL (2)*. The Association for Computer Linguistics.

Verónica Pérez-Rosas, Bennett Kleinberg, Alexandra Lefevre, and Rada Mihalcea. 2018. Automatic Detection of Fake News. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3391–3401. Association for Computational Linguistics.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep Contextualized Word Representations. In *NAACL-HLT*, pages 2227–2237. Association for Computational Linguistics.

Dean Pomerleau and Delip Rao. 2017. The Fake News Challenge: Exploring how artificial intelligence technologies could be leveraged to combat fake news. `http://www.fakenewschallenge.org/`. Accessed: 2019-02-14.

Kashyap Popat, Subhabrata Mukherjee, Jannik Strötgen, and Gerhard Weikum. 2016. Credibility Assessment of Textual Claims on the Web. In *CIKM*, pages 2173–2178.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know What You Don't Know: Unanswerable Questions for SQuAD. In *ACL (2)*, pages 784–789. Association for Computational Linguistics.

Benjamin Riedel, Isabelle Augenstein, Georgios P. Spithourakis, and Sebastian Riedel. 2017. A simple but tough-to-beat baseline for the Fake News Challenge stance detection task. *CoRR*, abs/1707.03264.

Victoria Rubin, Niall Conroy, Yimin Chen, and Sarah Cornwell. 2016. Fake News or Truth? Using Satirical Cues to Detect Potentially Misleading News. In *Proceedings of the Second Workshop on Computational Approaches to Deception Detection*, pages 7–17. Association for Computational Linguistics.

Giovanni C Santia and Jake Ryland Williams. 2018. BuzzFace: A News Veracity Dataset with Facebook User Commentary and Egos. *ICWSM*, 531:540.

K. Shu, D. Mahudeswaran, S. Wang, D. Lee, and H. Liu. 2018. FakeNewsNet: A Data Repository with News Content, Social Context and Dynamic Information for Studying Fake News on Social Media. *ArXiv e-prints*.

Eugenio Tacchini, Gabriele Ballarin, Marco L. Della Vedova, Stefano Moret, and Luca de Alfaro. 2017. Some Like it Hoax: Automated Fake News Detection in Social Networks. In *Proceedings of the Second Workshop on Data Science for Social Good (SoGood)*, volume 1960 of *CEUR Workshop Proceedings*.

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018. FEVER: a Large-scale Dataset for Fact Extraction and VERification. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 809–819, New Orleans, Louisiana. Association for Computational Linguistics.

Andreas Vlachos and Sebastian Riedel. 2014. Fact Checking: Task definition and dataset construction. In *Proceedings of the ACL 2014 Workshop on Language Technologies and Computational Social Science*, pages 18–22. Association for Computational Linguistics.

Soroush Vosoughi, Deb Roy, and Sinan Aral. 2018. The spread of true and false news online. *Science*, 359(6380):1146–1151.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2018a. GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding. In *BlackboxNLP@EMNLP*, pages 353–355. Association for Computational Linguistics.

Dongsheng Wang, Jakob Grue Simonsen, Birger Larsen, and Christina Lioma. 2018b. The Copenhagen Team Participation in the Factuality Task of the Competition of Automatic Identification and Verification of Claims in Political Debates of the CLEF-2018 Fact Checking Lab. In *CLEF (Working Notes)*, volume 2125 of *CEUR Workshop Proceedings*. CEUR-WS.org.

William Yang Wang. 2017. "Liar, Liar Pants on Fire": A New Benchmark Dataset for Fake News Detection. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 422–426. Association for Computational Linguistics.

Wenpeng Yin and Dan Roth. 2018. TwoWingOS: A Two-Wing Optimization Strategy for Evidential Claim Verification. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 105–114, Brussels, Belgium. Association for Computational Linguistics.

Arkaitz Zubiaga, Elena Kochkina, Maria Liakata, Rob Procter, Michal Lukasik, Kalina Bontcheva, Trevor Cohn, and Isabelle Augenstein. 2018. Discourse-aware rumour stance classification in social media using sequential classifiers. *Informatino Processing & Management*, 54(2):273–290.

Arkaitz Zubiaga, Maria Liakata, Rob Procter, Geraldine Wong Sak Hoi, and Peter Tolmie. 2016. Analysing how people orient to and spread rumours in social media by looking at conversational threads. *PLOS ONE*, 11(3):1–29.

Websites (Sources)	Reason
Mediabiasfactcheck	Website that checks other news websites
CBC	No pattern to crawl
apnews.com/APFactCheck	No categorical label and no structured claim
weeklystandard.com/tag/fact-check	Mostly no label, and they are placed anywhere
ballotpedia.org	No categorical label and no structured claim
channel3000.com/news/politics/reality-check	No categorical label, lack of structure, and no clear claim
npr.org/sections/politics-fact-check	No label and no clear claim (only some titles are claims)
dailycaller.com/buzz/check-your-fact	Is a subset of checkyourfact which has already been crawled
sacbee.com[6]	Contains very few labelled articles, and without clear claims
TheGuardian	Only a few websites have a pattern for labels.

Table 9: The list of websites that we did not crawl and reasons for not crawling them.

Domain	# Insts	# Labels	Labels
abbc	436	3	in-between, in-the-red, in-the-green
afck	433	7	correct, incorrect, mostly-correct, unproven, misleading, understated, exaggerated
bove	295	2	none, rating: false
chct	355	4	verdict: true, verdict: false, verdict: unsubstantiated, none
clck	38	3	incorrect, unsupported, misleading
faan	111	3	factscan score: false, factscan score: true, factscan score: misleading
faly	71	5	true, none, partly true, unverified, false
fani	20	3	conclusion: accurate, conclusion: false, conclusion: unclear
farg	485	11	false, none, distorts the facts, misleading, spins the facts, no evidence, not the whole story, unsupported, cherry picks, exaggerates, out of context
goop	2943	6	0, 1, 2, 3, 4, 10
hoer	1310	7	facebook scams, true messages, bogus warning, statirical reports, fake news, unsubstantiated messages, misleading recommendations
huca	34	3	a lot of baloney, a little baloney, some baloney
mpws	47	3	accurate, false, misleading
obry	59	4	mostly_true, verified, unobservable, mostly_false
para	222	7	mostly false, mostly true, half-true, false, true, pants on fire!, half flip
peck	65	3	false, true, partially true
pomt	15390	9	half-true, false, mostly true, mostly false, true, pants on fire!, full flop, half flip, no flip
pose	1361	6	promise kept, promise broken, compromise, in the works, not yet rated, stalled
ranz	21	2	fact, fiction
snes	6455	12	false, true, mixture, unproven, mostly false, mostly true, miscaptioned, legend, outdated, misattributed, scam, correct attribution
thet	79	6	none, mostly false, mostly true, half true, false, true
thal	74	2	none, we rate this claim false
tron	3423	27	fiction!, truth!, unproven!, truth! & fiction!, mostly fiction!, none, disputed!, truth! & misleading!, authorship confirmed!, mostly truth!, incorrect attribution!, scam!, investigation pending!, confirmed authorship!, commentary!, previously truth! now resolved!, outdated!, truth! & outdated!, virus!, fiction! & satire!, truth! & unproven!, misleading!, grass roots movement!, opinion!, correct attribution!, truth! & disputed!, inaccurate attribution!
vees	504	4	none, fake, misleading, false
vogo	653	8	none, determination: false, determination: true, determination: mostly true, determination: misleading, determination: barely true, determination: huckster propaganda, determination: false, determination: a stretch
wast	201	7	4 pinnochios, 3 pinnochios, 2 pinnochios, false, not the whole story, needs context, none

Table 10: Number of instances, and labels per domain sorted by number of occurrences

Website	Domain	Claims	Labels	Category	Speaker	Checker	Tags	Article	Claim date	Publish date	Full text	Outlinks
abc	abbc	436	436	436	-	-	436	436	-	436	436	7676
africacheck	afck	436	436	-	-	-	-	436	-	436	436	2325
altnews	-	496	-	-	-	496	-	496	-	496	496	6389
boomlive	-	302	302	-	-	-	-	302	-	302	302	6054
checkyourfact	chht	358	358	-	-	358	-	-	-	358	358	5271
climatefeedback	clck	45	45	-	-	-	-	45	-	45	45	489
crikey	-	18	18	18	-	18	18	18	-	18	18	212
factcheckni	-	36	36	36	-	-	-	36	-	-	36	151
factcheckorg	farg	512	512	512	512	512	512	512	512	512	512	8282
factly	-	77	77	-	-	-	-	77	-	-	77	658
factscan	-	115	115	-	115	-	-	-	115	115	115	1138
fullfact	-	336	336	336	-	336	-	336	-	336	336	3838
gossipcop	goop	2947	2947	-	-	2947	-	2947	-	2947	2947	12583
hoaxslayer	hoer	1310	1310	-	-	1310	-	1310	-	1310	1310	14499
huffingtonpostca	huca	38	38	-	38	38	-	38	38	38	38	78
leadstories	-	1547	1547	-	-	1547	-	1547	-	1547	1547	12015
mprnews	mpws	49	49	-	-	49	-	49	-	49	49	319
nytimes	-	17	17	-	-	17	-	17	-	17	17	271
observatory	obry	60	60	-	-	60	-	60	-	60	60	592
pandora	para	225	225	225	225	225	-	225	-	225	225	114
pesacheck	peck	67	67	-	-	67	-	67	-	67	67	521
politico	-	102	102	-	-	102	-	102	-	102	102	150
politifact_promise	pose	1361	1361	1361	1361	-	-	1361	-	1361	1361	6279
politifact_stmt	pomt	15390	15390	-	15390	-	-	-	15390	15390	15390	78543
politifact_story	-	5460	-	-	-	5460	-	-	-	5460	5460	24836
radionz	ranz	32	32	32	32	-	-	32	32	32	32	44
snopes	snes	6457	6457	6457	-	6457	-	6457	-	6457	6457	46735
swissinfo	-	20	20	20	20	20	-	20	-	20	20	40
theconversation	-	62	62	62	62	62	62	62	-	62	62	723
theferret	thet	81	81	81	81)	-	-	81	-	81(81)	81	885
theguardian	-	155	155	155	-	155	-	155	-	155	155	2600
thejournal	thal	179	179	-	-	-	-	179	-	179	179	2375
truthorfiction	tron	3674	3674	3674	-	-	3674	3674	-	3674	3674	8268
verafiles	vees	509	509	-	-	-	509	509	-	509	509	23
voiceofsandiego	vogo	660	660	-	-	-	-	660	-	660	660	2352
washingtonpost	wast	227	227	-	227	227	-	227	-	227	227	2470
wral	-	20	20	-	-	20	20	20	-	20	20	355
zimfact	-	21	21	21	21	21	-	21	-	21	21	179
Total		43837	43837	43837	43837	43837	43837	43837	43837	43837	43837	260330

Table 11: Summary statistics for claim collection. 'Domain' indicates the domain name used for the veracity prediction experiments, '–' indicates that the website was not used due to missing or insufficient claim labels, see Section 3.2.

Chapter 6

Factuality Checking in News Headlines with Eye Tracking

Christian Hansen, Casper Hansen, Jakob Grue Simonsen, Birger Larsen, Stephen Alstrup, Christina Lioma (2020). Factuality Checking in News Headlines with Eye Tracking. In SIGIR, pages 2013-2016. [43].

Factuality Checking in News Headlines with Eye Tracking

Christian Hansen
University of Copenhagen
chrh@di.ku.dk

Casper Hansen
University of Copenhagen
c.hansen@di.ku.dk

Jakob Grue Simonsen
University of Copenhagen
simonsen@di.ku.dk

Birger Larsen
Aalborg University
birger@hum.aau.dk

Stephen Alstrup
University of Copenhagen
s.alstrup@di.ku.dk

Christina Lioma
University of Copenhagen
c.lioma@di.ku.dk

ABSTRACT

We study whether it is possible to infer if a news headline is true or false using only the movement of the human eyes when reading news headlines. Our study with 55 participants who are eye-tracked when reading 108 news headlines (72 true, 36 false) shows that false headlines receive statistically significantly less visual attention than true headlines. We further build an ensemble learner that predicts news headline factuality using only eye-tracking measurements. Our model yields a mean AUC of 0.688 and is better at detecting false than true headlines. Through a model analysis, we find that eye-tracking 25 users when reading 3-6 headlines is sufficient for our ensemble learner.

KEYWORDS

Factuality checking; Eye tracking; Fake news

ACM Reference Format:
Christian Hansen, Casper Hansen, Jakob Grue Simonsen, Birger Larsen, Stephen Alstrup, and Christina Lioma. 2020. Factuality Checking in News Headlines with Eye Tracking. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '20), July 25–30, 2020, Virtual Event, China.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3397271.3401221

1 INTRODUCTION AND PRIOR WORK

Factuality detection in headlines is important because headlines are often solely responsible for the user's first impression (especially in mobile environments); but it is also challenging because, unlike full text, news headlines convey information succinctly and without reasoned argumentation or background.

We measure the overt attention of 55 participants who are eye-tracked when reading 108 news headlines. We find statistically significantly longer eye gazing and fixation durations when reading headlines of true, rather than false news, regardless of participant gender. We also train an ensemble learner, solely on eye-tracking data, to infer factuality in headlines. Our model yields a mean AUC of 0.688 and is better at detecting false headlines than true headlines. Further analysis shows that eye-tracking 25 users when reading 3-6 headlines is sufficient for our ensemble learner.

Eye tracking has long been used in IR to infer relevance [1, 3, 4, 7, 8, 11] and to improve user understanding, for instance that adding information to search engine snippets significantly improves performance for informational tasks but degrades performance for navigational tasks [5]; that users with higher change in knowledge differ significantly in terms of the number and duration of fixations compared to users with lower knowledge-change [2]; and that relevant documents tend to be continuously read, while irrelevant documents tend to be scanned [6]. In most cases, cognitive effort inferred from eye-tracking data is highest for (at least) partially relevant documents and lowest for irrelevant documents.

Our findings complement prior findings that news posts from credible sources receive more gaze attention [13] and that false news tend to be read more quickly than accurate news [6]. However, none of the above studies is done on headlines, and, to our knowledge, we present the first factuality inference model to be trained exclusively on eye-tracked data.

2 EXPERIMENT DESIGN

55 participants with normal or corrected-to-normal vision were recruited (24 females, 31 males; 19-33 years of age, median age 24), and each participated in a single eye tracking session in a laboratory. At the start of each session, we logged the age and gender of each participant and then introduced the task and apparatus. The eye tracker was calibrated and the task commenced. On completion of the task, participants were debriefed and comments were solicited. At no time were participants informed about how well they were doing. Each participant was shown a screen (white background) with three headlines (each on a separate line, in black font, size=36), without any further information. The headlines were centered on the screen, with 70mm of space between them and 20mm of space to the left border of the screen. Participants were asked to choose the most recent headline. This task was chosen on purpose to keep participants engaged in reading under circumstances where they were not directly checking for factuality. When participants had made their choice, the next screen (showing three new headlines) appeared. Participants did not know that two of the headlines were true and one was false, at any time. In total, 36 screens, each with three different headlines, were shown (108 unique headlines). To address order effects, we fully counterbalanced the position (top, middle, bottom) of the headlines, so that each position contained a factually false headline exactly 12 times. Participants could not move on to the next screen before answering, with no possibility of giving a "don't know"-answer, and could not revisit a previous

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

SIGIR '20, July 25–30, 2020, Virtual Event, China

© 2020 Copyright held by the owner/author(s). Publication rights licensed to ACM.

ACM ISBN 978-1-4503-8016-4/20/07…$15.00
https://doi.org/10.1145/3397271.3401221

Table 1: Dataset statistics.

	True	False	Total
# Headlines	72	36	108
Mean # words per headline	8.56	8.42	8.51
Mean # content words per headline	4.79	4.53	4.70
Mean # function words per headline	3.88	4.08	3.95

Table 2: All transformations that falsified news headlines.

original text	transformed text
more, most, best, top, highest, good	fewer, least, worst, bottom, lowest, bad
denies, fear, pick up award, react to	admits, love, stripped of award, praise
two ... in top 50, remain, helping out	no ... in top 50, exit, refuses to help
criticised, leads in, drops down	praised, last in, tops
cannot get enough of, calls for end	do no like, tolerates
looks to . . . as inspiration	uses . . . as example to avoid

screen. All participants saw the same 36 screens with the order of screens randomized across participants. No time limit was set for completing the task.

To calibrate the experimental design, we did a pre-study on 11 participants with a subset of 24 screens. The pre-study did not lead to any changes in the design or protocol, except that the number of screens was increased to 36 because participants were faster than initially expected. In our analysis we combine the data from the pre-study with the remaining data to form the complete dataset.

Each participant performed the task individually, and was given the same oral instructions by the research assistant[1]. Participants could at all times elect to stop the experiment (none did). The study was approved by the ethics board of our university, and all data was anonymized prior to storage and analysis.

The headlines shown to participants were crawled from the website of a reputable local newspaper[2] and consisted of the full title of an article concerning *local and national news*. From the pool of crawled headlines, we selected 108 headlines that: (a) covered news that should be generally known to the public, (b) were formulated in approximately the same tone (i.e., no clickbait titles, no emphatics, no puns), and (c) were unlikely to provoke strong feelings. All headlines were selected manually by one of the authors of this paper (see Table 1 for their statistics).

All crawled headlines were factually true. We created factually false headlines by semantically reversing parts of some headlines. For example, `··· among` **`most`** `expensive cities to relocate` to became `··· among` **`least`** `expensive cities to relocate to`. All the semantic transformations we used to falsify headlines are shown in Table 2. When falsifying headlines, we made sure that they still appeared semantically plausible and sounded natural. To make sure that there is no bias stemming from the linguistic formulation of true versus false headlines, we POS-tagged all headlines (using the Stanford parser) and found that the proportion of content words (which are known to be fixated on by the human eye much more than functions words [12]) was approximately the same in both true and false headlines (see Table 1). We make all 108 headlines freely available[1].

Apparatus. We used an Eyetribe ET1000 desk-mounted stream-based eye tracker bar, paired with a 24-inch screen (resolution of 1920x1200 and 170 DPI). The eye tracker sampled the position of eyes at the rate of 30 Hz and had a spatial resolution of 0.1 degree. We used iMotions[3] to calibrate the eye tracker and collect the data. Participants were placed 60cm away from the screen, and the room had soft standard artificial light. No head stabilisation was used (head movements were unconstrained so the intrusion of the eye moving measurement was minimal). We calibrated the eye tracker using a standard 9-point calibration prior to each recording.

Participants indicated which of the three headlines per screen was the most recent by typing 1, 2, or 3 on the keyboard (for the position of the top, middle, and respectively bottom headline). Typing was chosen over using the cursor because the cursor could interfere considerably with eye tracking.

Eye-tracking measures. A fixation is a stable eye-in-head position within a dispersion threshold (typically 2 degrees), above a duration threshold (typically 100-200 milliseconds[4]), and velocity below a threshold (typically 15-100 degrees per second). Gaze duration is the cumulative duration of a sequence of consecutive fixations within an area of interest (AOI). We defined a separate AOI around each headline and we analysed these 5 measures: the total time spent fixating inside an AOI (**total fixation duration**); the total number of fixations inside a AOI (**total fixation count**; the total time spent gazing inside an AOI (**total gaze duration**)[5]; the **average fixation duration** inside an AOI (total fixation duration divided by total fixation count); the duration of the first fixation inside an AOI (**first fixation duration**).

3 FINDINGS

We now study the statistical effect the headline factuality has on the eye-tracking measures. Let γ denote any of the above 5 eye-tracking measures. To establish whether factuality affects each of these γs in a statistically significant way, we consider both fixed effects (gender, headline length, position of headline on screen), and random effects. These fixed and random effects are potentially non-negligible, meaning that conventional methods for inferential data analysis, such as ANOVA and general linear regression are not applicable [8]. We therefore fit a mixed model [15] that uses the above γs as a response and the fixed effects as explanatory variables. Because each participant is drawn from some larger population, the participant is included as a random intercept. The mixed model for each of the above γs is:

$$\gamma = c_{\text{true}} i_{\text{true}} + c_{\text{middle}} i_{\text{middle}} + c_{\text{bottom}} i_{\text{bottom}} + c_{\text{male}} i_{\text{male}} + c_{\text{length}} l + p + b$$

where c_{factor} is the coefficient for the factor and i_{factor} is the indicator function for the factor, e.g. $i_{\text{male}} = 1$ if the participant is male and $i_{\text{male}} = 0$ otherwise. For the categorical variables of position (middle, bottom), gender (male), and factuality (true), there are $k - 1$ fewer factors than number of categories (k). l is the normalised length of the headline with zero mean and unit variance, p is the random effect for the participant, and b is the intercept. The model is fitted using the γs collected; these γs are normalised so that the scale of the coefficient is comparable across measures, which otherwise have different scales.

[1] https://github.com/Varyn/Factuality_Checking_News_Headlines_EyeTracking

[2] https://www.thelocal.dk/

[3] https://imotions.com/

[4] We set fixations at 100 milliseconds.

[5] Gaze duration consists of the duration of fixations and other captured gaze activity (such as time between fixations) inside an AOI.

The coefficient c_{true} shows the relation between the measure γ and the factuality of the headline. We formulate the null hypothesis H_0^γ for γ as the assumption that factuality does not affect γ, that is $H_0^\gamma : c_{\text{true}} = 0$. To test this hypothesis, we compute p-values and confidence intervals for each coefficient by performing Wald tests. We have 5 different eye-tracking measures, so we perform 5 hypothesis tests with Bonferroni correction, requiring that $p < \frac{0.05}{5} = 0.01$ to reject each H_0^γ. All statistical analysis is done using StatsModels[6], and the models are fitted using Maximum Likelihood.

Table 3 shows the resulting coefficients. We see that for *total gaze duration, total fixation duration*, and *total fixation count* $p < .001$, thus we have sufficient evidence to reject the null hypothesis. These three eye-tracking measures change significantly when reading true versus false headlines. However, for *average fixation duration* and *first fixation duration*, we cannot reject the null hypothesis, and thus we cannot conclude that the time spent on each individual fixation changes between factually true and false headlines. We also observe that a factually true headline causes the *total gaze duration, total fixation duration*, and *total fixation count* to increase, as seen by the positive value of c_{true}; this means that false headlines in general have shorter fixation and gazing duration than true headlines. The fact that factuality is not significant for *average fixation duration* means that the increased *total fixation duration* for true headlines is caused by an increase in *total fixation count* for factually true headlines.

We now briefly discuss the other coefficients than c_{true}. Using $p < 0.01$, we see that the position of the headline is not significant for the *total gaze duration*, while it is significant if the headline is placed on the bottom for all measures of fixation. The negative value of c_{bottom} shows that all measures of fixation decrease when the headline is placed on the bottom. The length of the headline is significant for all eye-tracking measures (p < 0.001), with longer headlines having higher measures. Lastly, we observe no significant difference in any measures between the genders.

Table 3: The fixed effects for the five eye-tracking measures. p-values below 0.01 are marked in bold. (See Section 3 for notation).

	Coef.	Std.Err.	z	P>|z|	[0.025	0.975]
Total gaze duration						
c_{true}	0.154	0.023	6.697	**<0.001**	0.109	0.199
c_{middle}	-0.026	0.027	-0.959	0.338	-0.078	0.027
c_{bottom}	-0.054	0.027	-2.020	0.043	-0.106	-0.002
c_{male}	-0.149	0.154	-0.969	0.333	-0.451	0.153
c_{length}	0.174	0.011	15.844	**<0.001**	0.153	0.196
Total fixation duration						
c_{true}	0.109	0.021	5.301	**<0.001**	0.069	0.149
c_{middle}	-0.083	0.024	-3.474	**<0.001**	-0.129	-0.036
c_{bottom}	-0.239	0.024	-10.059	**<0.001**	-0.285	-0.192
c_{male}	-0.202	0.182	-1.109	0.267	-0.558	0.155
c_{length}	0.100	0.010	10.154	**<0.001**	0.081	0.119
Total fixation count						
c_{true}	0.115	0.020	5.609	**<0.001**	0.075	0.155
c_{middle}	-0.037	0.024	-1.536	0.124	-0.083	0.010
c_{bottom}	-0.199	0.024	-8.420	**<0.001**	-0.246	-0.153
c_{male}	-0.164	0.184	-0.894	0.371	-0.524	0.196
c_{length}	0.118	0.010	12.011	**<0.001**	0.099	0.137
Average fixation duration						
c_{true}	0.025	0.022	1.106	0.269	-0.019	0.068
c_{middle}	-0.003	0.026	-0.125	0.900	-0.054	0.047
c_{bottom}	-0.130	0.026	-5.061	**<0.001**	-0.181	-0.080
c_{male}	-0.006	0.171	-0.038	0.970	-0.342	0.329
c_{length}	0.059	0.011	5.509	**<0.001**	0.038	0.079
First fixation duration						
c_{true}	0.034	0.024	1.411	0.158	-0.013	0.081
c_{middle}	0.014	0.028	0.484	0.628	-0.041	0.068
c_{bottom}	-0.120	0.028	-4.321	**<0.001**	-0.175	-0.066
c_{male}	-0.016	0.148	-0.106	0.915	-0.305	0.274
c_{length}	0.056	0.011	4.906	**<0.001**	0.034	0.079

Learning to infer factuality from eye tracking. Having established that *total gaze duration, total fixation duration*, and *total fixation count* are all significantly different depending on the headline factuality, we next investigate if these measures provide sufficient signal for training a headline factuality classifier. As these measures are highly dependent on the length and position of the headlines, they are also included in the model. We observe that *total fixation duration* is highly correlated with *total fixation count*, thus to keep the model as simple as possible, we only use *total gaze duration* and *total fixation duration*.

In table 3, we see the coefficient of factuality (c_{true}), for many measures, to be less influential than the position and length of the headline. Thus, we expect using eye-tracking measures of only a single participant to be noisy. Due to this, we use an ensembling approach, where the predicted factuality of a headline is computed as an average over a set of participants (P_{ens}): $v_h = \frac{1}{|P_{\text{ens}}|} \sum_{p \in P_{\text{ens}}} v_{p,h}$, where v_h is the factuality prediction for headline h, and $v_{p,h}$ is the factuality prediction for headline h for participant p. Due to the relative small size of our dataset, we propose to use the average of two simple second-order logistic models for estimating $v_{p,h}$:

$$v^1_{p,h} = \frac{1}{1 + e^{-(c_1 i_{\text{top}}(h)\gamma^{\text{GD}}_{p,h} + c_2 i_{\text{middle}}(h)\gamma^{\text{GD}}_{p,h} + c_3 i_{\text{bottom}}(h)\gamma^{\text{GD}}_{p,h})}} \tag{1}$$

$$v^2_{p,h} = \frac{1}{1 + e^{-(c_4 l_h \gamma^{\text{FD}}_{p,h})}}, \quad v_{p,h} = \frac{v^1_{p,h} + v^2_{p,h}}{2} \tag{2}$$

where $c_k, k \in [1, 2, 3, 4]$ are the learned coefficients of the logistic models, $\gamma^{\text{GD}}_{p,h}$ is the total gaze duration for participant p on headline h, $\gamma^{\text{FD}}_{p,h}$ is the total fixation duration for p on h, and l is the length of the headline. Both logistic models have one eye-tracking measure interacting with either the length or position of the headline, where the interaction is chosen based on the pair with the lowest correlation. We choose to use two simple logistic models, instead of a single combined model, to increase the variance of the predicted factuality, as high variance is beneficial for ensembling. We standardize (zero mean and unit variance) the eye-tracking measures from each participant across all headlines. Lastly, the two logistic models are trained using Maximum Likelihood on a set of training participants.

Evaluation. We evaluate the model by inferring factuality on unseen headlines using Monte Carlo cross-validation over 100,000 iterations. In each iteration, the participants are split for training and ensembling (27 and 28 participants, respectively), and three headlines are chosen for evaluation (2 true and 1 false), while the remaining headlines are used for training. We report the mean AUC and mean accuracy, across all iterations.

[6] https://www.statsmodels.org/stable/index.html, version 0.9

Table 4: Factuality performance scores from our eye-tracking ensemble model.

Mean AUC	Mean Acc.	Mean Acc. (True)	Mean Acc. (False)
0.688	0.634	0.619	0.662

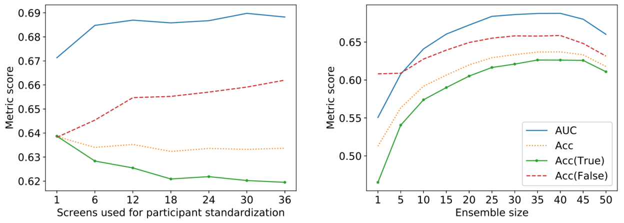


Figure 1: Performance analysis when varying (left) the number of screens used for participant standardization in our model and (right) the number of participants used for ensembling.

As reported in Table 4, we find that our ensemble model predicts the factuality of unseen headlines with a mean AUC of 0.688 and an accuracy of 0.634 (which is higher on false headlines (0.662) than one true ones (0.619)). There is no prior work on automatically detecting factuality in news headlines only, but related work on inferring factuality in text (but not headlines, which is harder) using textual features alone (not eye-tracking features), shows that accuracy ranges from 0.39 [14] to 0.76 [9], and even up to 0.86 [10] when using BiLSTMs and a multilayer perceptron classifier with refined linguistic features such as entailment and contradiction. Comparably, we have a simple learning model, which uses weaker input features (eye-tracking measures are less discriminative than textual ones), and which solves a more difficult problem (factuality checking in headlines instead of longer texts).

Analysis. In the above, we standardize the eye-tracking measures for each participant on all headlines. We now ask: how important is this standardization, and would standardization on fewer headlines suffice? We answer this by sampling fewer headlines to base the standardization on, while still preserving the ratio of 2 true headlines for each false one. We refer to three headlines following this ratio as a "screen".

Figure 1 (left) shows the mean accuracy and AUC when varying the number of screens used for standardization. When only standardizing on the screen we predict on (screen=1), mean AUC is at minimum; it drastically increases at 6 screens, and then stabilizes for the remaining number of screens. When increasing the number of screens, the accuracy for the true headlines decreases slightly, while the accuracy increases for the false headlines, but after 6-18 screens the difference of including more screens is minimal. This suggests that the performance of our ensemble model is not largely dependent on a large set of headlines to use for standardization. Deployed on a live setup, few headlines for standardization could suffice to fetch the accuracy and AUC levels reported in this study.

The results reported above correspond to splitting participants approximately 50/50 for training and ensembling, and this split can of course be varied; Figure 1 (right) plots mean accuracy and AUC (y axis) across a varying number of participants used for ensembling out of the 55 participants in total. We see that the choice of a 50/50 split is close to optimal. The fact that performance drops rapidly when 15 or fewer participants are used for ensembling indicates that aggregating over a large set of participants is at least as important as training a model on more data, in this setup. This happens because our dataset is small (we have few participants), so the optimal performance is a trade-off between training a better model (requiring more participants for training) and aggregating over more participants (requiring more participants in the ensemble).

4 CONCLUSIONS

We studied whether the human eye moves differently when reading factually true versus factually false news headlines, and if we can infer factuality in news headlines using only eye-tracking signals. In an experiment with 55 users reading 108 news headlines, we found that false headlines receive statistically significantly less visual attention than true ones. We used this to build an ensemble learner that predicts news headline factuality using only eye-tracking measurements, which obtained a mean AUC score of 0.688 and a mean accuracy of 0.634.

Future work includes investigation of eye tracking as a boosting mechanism to potentially improve factuality detection based on text processing, and refining the relationship between eye movements in more typical IR tasks such as search. A different direction of promising future work is to repeat our study "in the wild" outside usual laboratory settings, including eye-tracking methods with lower fidelity, such as for instance typical cameras mounted on laptops and smartphone cameras.

REFERENCES

[1] Antti Ajanki, David R. Hardoon, Samuel Kaski, Kai Puolamäki, and John Shawe-Taylor. 2009. Can eyes reveal interest? Implicit queries from gaze patterns. *User Model. User-Adapt. Interact.* 19, 4 (2009), 307–339.

[2] Nilavra Bhattacharya and Jacek Gwizdka. 2018. Relating eye-tracking measures with changes in knowledge on search tasks. In *ETRA*. 62:1–62:5.

[3] Georg Buscher, Andreas Dengel, Ralf Biedert, and Ludger V. Elst. 2012. Attentive Documents: Eye Tracking As Implicit Feedback for Information Retrieval and Beyond. *ACM Trans. Interact. Intell. Syst.* 1, 2, Article 9 (2012), 30 pages.

[4] Georg Buscher, Andreas Dengel, and Ludger van Elst. 2008. Query expansion using gaze-based feedback on the subdocument level. In *SIGIR*. 387–394.

[5] Edward Cutrell and Zhiwei Guan. 2007. What are you looking for?: an eye-tracking study of information usage in web search. In *CHI*. 407–416.

[6] Jacek Gwizdka. 2014. News stories relevance effects on eye-movements. In *ETRA*. 283–286.

[7] David Hardoon, John Shawe-Taylor, Antti Ajanki, Kai Puolamäki, and Samuel Kaski. 2007. Information Retrieval by Inferring Implicit Queries from Eye Movements. *Journal of Machine Learning Research - Proceedings Track* 2 (12 2007), 179–186.

[8] Tomasz D. Loboda, Peter Brusilovsky, and Jörg Brunstein. 2011. Inferring word relevance from eye-movements of readers. In *IUI*. 175–184.

[9] Verónica Pérez-Rosas, Bennett Kleinberg, Alexandra Lefevre, and Rada Mihalcea. 2018. Automatic Detection of Fake News. In *COLING*. 3391–3401.

[10] Adam Poliak, Jason Naradowsky, Aparajita Haldar, Rachel Rudinger, and Benjamin Van Durme. 2018. Hypothesis Only Baselines in Natural Language Inference. In *Proceedings of the Seventh Joint Conference on Lexical and Computational Semantics*. Association for Computational Linguistics, New Orleans, Louisiana, 180–191.

[11] Kai Puolamäki, Antti Ajanki, and Samuel Kaski. 2008. Learning to learn implicit queries from gaze patterns. In *ICML*. 760–767.

[12] Keith Rayner. 1998. Eye movements in reading and information processing: 20 years of research. *Psychological Bulletin* 124, 3 (1998), 372–422.

[13] Michael Sülflow, Svenja Schäfer, and Stephan Winter. 2019. Selective attention in the news feed: An eye-tracking study on the perception and selection of political news posts on Facebook. *New Media & Society* 21, 1 (2019).

[14] William Yang Wang. 2017. "Liar, Liar Pants on Fire": A New Benchmark Dataset for Fake News Detection. In *ACL*. 422–426.

[15] Lang Wu. 2009. *Mixed Effect Models for Complex Data*. Monographs on Statistics and Applied Probability, Vol. 113. CRC Press.

Chapter 7

Neural Check-Worthiness Ranking with Weak Supervision: Finding Sentences for Fact-Checking

Casper Hansen, Christian Hansen, Stephen Alstrup, Jakob Grue Simonsen, Christina Lioma (2019). Neural Check-Worthiness Ranking with Weak Supervision: Finding Sentences for Fact-Checking. In Companion Proceedings of WWW, pages 994-1000. [29].

Neural Check-Worthiness Ranking with Weak Supervision: Finding Sentences for Fact-Checking

Casper Hansen, Christian Hansen, Stephen Alstrup, Jakob Grue Simonsen, Christina Lioma
Department of Computer Science, University of Copenhagen

ABSTRACT

Automatic fact-checking systems detect misinformation, such as fake news, by (i) selecting *check-worthy* sentences for fact-checking, (ii) gathering related information to the sentences, and (iii) inferring the factuality of the sentences. Most prior research on (i) uses hand-crafted features to select check-worthy sentences, and does not explicitly account for the recent finding that the top weighted terms in both check-worthy and non-check-worthy sentences are actually overlapping [15]. Motivated by this, we present a neural check-worthiness sentence ranking model that represents each word in a sentence by *both* its embedding (aiming to capture its semantics) and its syntactic dependencies (aiming to capture its role in modifying the semantics of other terms in the sentence). Our model is an end-to-end trainable neural network for check-worthiness ranking, which is trained on large amounts of unlabelled data through weak supervision. Thorough experimental evaluation against state of the art baselines, with and without weak supervision, shows our model to be superior at all times (+13% in MAP and +28% at various Precision cut-offs from the best baseline with statistical significance). Empirical analysis of the use of weak supervision, word embedding pretraining on domain-specific data, and the use of syntactic dependencies of our model reveals that check-worthy sentences contain notably more identical syntactic dependencies than non-check-worthy sentences.

KEYWORDS

Fact checking; Check worthiness; Deep learning; Weak supervision.

ACM Reference Format:
Casper Hansen, Christian Hansen, Stephen Alstrup, Jakob Grue Simonsen, Christina Lioma. 2019. Neural Check-Worthiness Ranking with Weak Supervision: Finding Sentences for Fact-Checking. In *Companion Proceedings of the 2019 World Wide Web Conference (WWW'19 Companion), May 13–17, 2019, San Francisco, CA, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3308560.3316736

1 INTRODUCTION

The fast and wide spread of misinformation (as opposed to true information) on social media [22, 25], and the increasing use of social media as a source of news[1] has turned "fake news" into an important societal problem on a scale that requires automated solutions. An automated fact-checking [21] pipeline typically consists of three steps: (i) selecting *check-worthy* sentences (i.e. sentences that contain check-worthy claims and should be fact-checked), (ii) gathering related information to those sentences, and (iii) using this related information to infer the factuality of each check-worthy sentence. Prior research has so far focused mainly on steps (ii) and (iii), for instance by generating claim-specific queries and querying search engines for relevant supporting information [12], focusing on specific sources such as Twitter [1], or exploiting knowledge graphs from e.g. Wikipedia [5]. These approaches assume an input of check-worthy claims. Considerably less research has focused on detecting such check-worthy claims, that is, determining not whether a sentence is true or not, but whether a sentence contains a check-worthy claim (and should be fact-checked) or not.

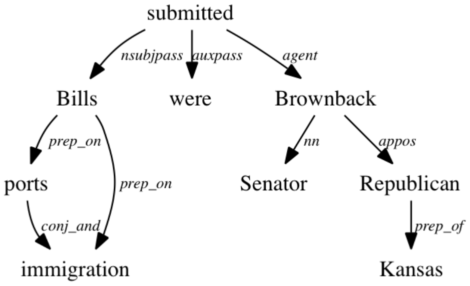


Figure 1: Syntactic dependencies example (from [20]).

Most research on check-worthiness detection uses hand-crafted features, such as bag-of-word representations, sentiment, and embedding averages [7, 8, 10, 19]. In addition, most work in this area does not explicitly account for the recent finding that the top weighted terms in both check-worthy and non-check-worthy sentences are actually overlapping [15], hence compromising the effectiveness of bag-of-word based methods.

Motivated by the above, we present a neural check-worthiness sentence ranking model that uses a dual sentence representation: each word in a sentence is represented by *both* its embedding (aiming to capture the semantics of that word from its context) and its syntactic dependencies (aiming to capture the role of that word in modifying the semantics of other words in the sentence, see Figure 1). We train the network with these dual representations end-to-end. This allows to learn such descriptive features directly from the input data, rather than relying on predetermined hand-crafted features that may not be useful for the task, and hence to adapt the representations to the domain. To tackle the problem of having very little available training data, we use an existing check-worthiness system to weakly label sentences, and we use this weakly labelled dataset to pretrain our neural network. This is inspired by recent strong results [18, 23] in various information retrieval tasks with few labelled data, but large amounts of unlabelled data.

[1] https://www.reuters.com/article/us-usa-internet-socialmedia/two-thirds-of-american-adults-get-news-from-social-media-survey-idUSKCN1BJ2A8

This paper is published under the Creative Commons Attribution 4.0 International (CC-BY 4.0) license. Authors reserve their rights to disseminate the work on their personal and corporate Web sites with the appropriate attribution.

WWW '19 Companion, May 13–17, 2019, San Francisco, CA, USA

© 2019 IW3C2 (International World Wide Web Conference Committee), published under Creative Commons CC-BY 4.0 License.

ACM ISBN 978-1-4503-6675-5/19/05.
https://doi.org/10.1145/3308560.3316736

Thorough experimental evaluation against all state of the art baselines on political speeches from the 2016 U.S. election, shows our model to be superior in all comparisons (+13% in MAP and up to +28% at various Precision cut-offs from the best baseline, with statistical significance). We empirically trace this superior performance to the use of syntactic dependencies in the sentence representation, where we find check-worthy sentences to contain notably more identical syntactic dependencies than non-check-worthy sentences. Further analysis shows that the performance benefits of weak supervision increase with the amount of data used, and that embeddings trained on smaller domain-specific data, as opposed to general purpose embeddings trained on the larger Google News corpus, increase effectiveness. In addition, despite using deep learning (a family of models that is generally considered of low interpretability [24]), the attention weighting used by our model on a word level provides humanly interpretable output, where the parts of the sentence that are important for the check-worthiness prediction can be determined.

We **contribute** a competitive and interpretable end-to-end trainable neural network model for check-worthiness ranking, which uses a dual input representation of both word embeddings and syntactic dependencies. Weak supervision is used to pretrain the network on large amounts of unlabelled data.

2 RELATED WORK

Given a sentence (also referred to as statement) as input, ClaimBuster [8, 9] outputs its check-worthiness score by extracting a set of features (sentiment, statement length, Part-of-Speech (POS) tags, named entities, and tf-idf weighted bag-of-words), and training a SVM classifier on these features to predict check-worthiness. Patwari et al. [19] present a system called TATHYA that is based on similar features, but that also includes as contextual features sentences immediately preceding and succeeding the one being assessed, as well as certain hand-crafted POS patterns. Gencheva et al. [7] also extend the feature set used by ClaimBuster to include more contextual features, such as the sentence's position in the debate text, and whether the debate opponent is mentioned. The work by Gencheva et al. has been extended to both English and Arabic [10]. In the recent CLEF 2018 competition on check-worthiness detection [17], Zou et al. [26] came first by using a large set of features (similarly to the above mentioned work), and doing feature selection with both a χ^2-test and a linear SVM using a L1 regularizer.

Prior work on neural networks for check-worthiness has been done by Konstantinovskiy et al. [14], who use InferSent [6] to derive a universal neural sentence representation and then train a logistic regression classifier on top of that. Similarly to our model, this approach also uses neural sentence embeddings. However, unlike our model, this approach uses *universal* sentence representations, whereas we train our model *end-to-end* to learn the representations directly from the input data, making the learning domain-specific.

In the related domain of sentence factuality detection Jimenez and Li [11] present a neural approach with multiple word embeddings. They artificially generate additional non-factual sentences to be used for training to increase robustness. Similarly to ours, this work also presents neural approaches that combine multiple word representations in order to improve performance. However, whereas the infusion of artificially generated non-factual sentences by Jimenez and Li [11] allows weak supervision of a single class, we obtain weak labels independently of the type of sentence (i.e. not on a single class).

3 NEURAL CHECK-WORTHINESS SENTENCE RANKING

Given a set of sentences as input, the aim is to rank them in descending order of check-worthiness. In order to better differentiate between sentences of varying degree of check-worthiness, We cast this as a ranking problem, as opposed to assigning a binary output to each sentence, following prior work [7, 8, 10] . Note that any ranked output can be made binary using an appropriate threshold, in case a subsequent fact-checking pipeline requires binary labelled sentences.

Given a set of sentences to be ranked, our model learns an end-to-end trained representation of each sentence. We describe first the representation of each word in the sentence (Section 3.1), followed by the neural network architecture (Section 3.2).

3.1 Neural sentence representation

Our model uses a dual sentence representation: each word in a sentence is represented by *both* its embedding and by its syntactic dependencies. The word embedding aims to capture the semantics of that word from its context. Embeddings of this type are generally well understood and have been found effective for check-worthiness detection [14]. The syntactic dependencies of a word aim to capture the role of that word in modifying the semantics of other words in the sentence, for instance by being the subject or predicate of the sentence. We use a syntactic dependency parser [4] to map each word to its dependency (as a tag) in relation to the sentence structure, which we then represent as a one-hot-encoding. Dependency parsing is fast using state of the art tools (approximately 14,000 words per second) [4].

Our motivation is that syntactic dependencies may be important for discriminating between common overlapping top-weighted words in both check-worthy and non-check-worthy sentences. The existence of common overlapping top weighted words in check-worthy and non-check-worthy sentences is reported by Le et al. [15] (see Figure 2 of [15] for examples), and to our knowledge, is not explicitly addressed by any prior check-worthiness approach. We posit that while these common top weighted words may not be distinguishable by their word representation, they may be distinguishable by their syntactic role in the sentence.

3.2 Network architecture

Based on the above, each word in a sentence has two distinct encodings, that together represent the word. We use this double representation of each word as input to a recurrent neural network (RNN) with GRU [3] memory units. The output from each word in the RNN is aggregated using an attention mechanism computed as $\alpha_t = \frac{\exp(\text{score}(h_t))}{\sum_i \exp(\text{score}(h_i))}$, where h_t is the output of the GRU memory cell at time t, and $score(\cdot)$ is a learned function that maps the output to a scalar. The attention-weighted sum is fed to a fully connected layer, from which the output is predicted using a sigmoid activation

Dataset	#docs	#sents.	sents. length	unique words	mean label
Embed. tr.	15,059	609,322	16.66	86,244	-
Evaluation	7	2,602	14.04	3,694	0.05
Weakly lab.	161	37,732	16.53	13,314	0.24

Table 1: Statistics of the embedding training, evaluation, and weakly labelled datasets.The evaluation dataset uses binary labels, and the weakly labelled dataset continuous labels in the interval [0, 1]**.**

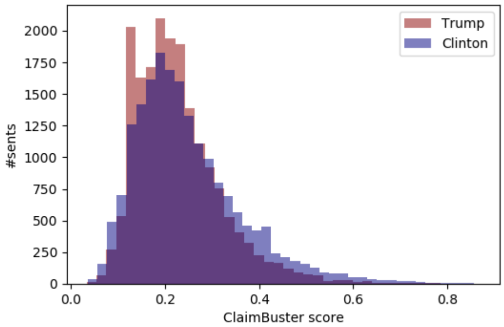


Figure 2: Histogram of the ClaimBuster scores used as the weak labels for each presidential candidate.

function. We train the network using the RMSprop optimizer with binary cross entropy as the loss function (details in Section 4.3).

4 EXPERIMENTAL SETUP

We conduct two experiments: (I) we compare our model against state of the art baselines without weak supervision; (II) we use the ClaimBuster API (one of the baselines in experiment I) to weakly label a dataset of unlabelled political speeches (as described in Section 4.2) and we use this weakly labelled data to pretrain the baselines and our model. ClaimBuster API is trained on a non-publicly available dataset and should therefore be considered as a black box baseline.

4.1 Baselines

We compare our model against these baselines (introduced in Section 2), which have yielded state of the art performance at their date of publication: (1) ClaimBuster and its pretrained ClaimBuster API [8], (2) TATHYA [19], and the approaches by (3) Zou et al. [26], (4) Gencheva et al. [7], and (5) Konstantinovskiy et al. [14].

4.2 Data

We use three datasets for three different purposes: (1) the *embedding training* dataset, used to train domain-specific embeddings for our model[2]; (2) the *evaluation* dataset, used to compare our model to the baselines without weak supervision; and (3) the *weakly labelled* dataset, used to compare our model to the baselines with weak supervision. We describe these next (see Table 1 for statistics).

The **Embedding Training Dataset** contains documents related to *all* U.S. elections available through the American Presidency Project[3], e.g. press releases, statements, speeches, and public fundraisers, resulting in 15,059 documents. We use this dataset to pretrain a domain specific word embedding for our model (see Section 5.2).

The **Evaluation Dataset** consists of a total of 2,602 sentences from 7 check-worthiness annotated political speeches from the 2016 U.S election. Out of these 7 speeches, 4 are by Donald Trump and are made available by the CLEF 2018 lab on automatic identification and verification of political claims [17]. The remaining are the inauguration and acceptance speech of Donald Trump and the acceptance speech of Hilary Clinton, and are made available by the authors of ClaimRank [10]. We choose the available PolitiFact annotated labels for this dataset.

The **Weakly Labelled Dataset** consists of all publicly available speeches by Hillary Clinton and Donald Trump from the 2016 U.S. election, which are available through the American Presidency Project. This amounts to 37,732 sentences from 161 speeches not occurring in the evaluation dataset. We use the public API[4] of ClaimBuster [8] to weakly label each sentence in all speeches. The ClaimBuster scores range from 0 to 1 (the higher the score, the more check-worthy the sentence), and are thus continuous as opposed to the binary labels from the evaluation dataset. The distribution of ClaimBuster scores can be seen in Figure 2, where we see that sentences by Hillary Clinton are overall labelled as slightly more check-worthy than those by Donald Trump.

4.3 Tuning

We measure the effectiveness of the ranking outputted by our model and the baselines using mean average precision (MAP), and average precision at multiple cut-offs: P@5, P@10, P@20, and P@R, where R is the number of check-worthy sentences in a given test set. We optimize the MAP when tuning parameters.

We tune and evaluate the approaches using 7-fold cross validation, where the sentences from one speech (see Section 4.2) act as testing data once, while sentences from the remaining speeches are used for training and validation. We use the sentences of each speech as input to the models. In all folds, we set aside 10% of the training data for validation. Each fold-wise evaluation is repeated 5 times with randomly chosen validation data. We report the average score of each metric across the folds and repetitions.

For ClaimBuster [9] and the model by Gencheva et al. [7], we use the best performing setup described in [7]: a double layered fully connected neural network with layer sizes of 200 and 50 respectively. We validate these sizes by keeping the same ratio (4:1) between the layers and test the largest sizes of $\{50, 100, 200, 400\}$, test batch sizes of $\{64, 128, 256, 512\}$, and use a learning rate of 0.0001. For the approach by Zou et al. [26] we use their multi-layer perceptron model with two hidden layers with sizes of 100 and 8 as per [26]. We validate these sizes by keeping the same ratio (12.5:1) between layers and test the largest sizes of $\{50, 100, 200, 400\}$ as done earlier. For TATHYA [19] we use the same multi-classifier approach and the same parameters as described in the original paper. For Konstantinovskiy et al. [14] we use the same logistic regression approach as described in the original paper.

For our model, we evaluate the same layer sizes as above with a ratio of 4:1 between the number of neurons in the GRU cell and the single fully connected layer. We train the word embeddings using the word2vec skip-gram model [16] on the embedding training dataset of 15,059 domain specific documents described in Section

[2] None of the other approaches support training embeddings.

[3] https://web.archive.org/web/20170606011755/http://www.presidency.ucsb.edu/

[4] https://idir-server2.uta.edu/claimbuster/

4.2. We use standard parameters with a window size of 5 and sample 25 negative samples per word. For the syntactic dependencies of each word, we use the spaCy syntactic dependency parser [4][5].

For the experiment with weak supervision, the ClaimBuster API returns a score from 0 to 1, indicating the degree of check-worthiness estimated by the system. In each fold in the 7-fold cross validation we find the threshold τ that splits the data and makes the fraction of check-worthy samples equal across the training without and with weakly labelled training data. Using these splits we evaluate two thresholding approaches: (1) Binarizing the labels based on τ, and (2) truncating all values larger than τ to the value of τ, and scaling the range $[0, \tau]$ into $[0, 1]$. In the cross validation, our approach performs best with step (2) and the baselines perform best with step (1) (these are the settings we report in Section 5). Note that step (2) can be considered as soft thresholding, as the labels are not binary. The weakly labelled data is used for pretraining the neural models or added to the training data for traditional supervised models.

5 RESULTS

Table 2 shows the results of the experimental comparison of our model to the baselines without and with weak supervision. We see that our model outperforms all baselines across all metrics (with improvements ranging from +9-28%), except P@5 (only without weak supervision) where our model is the second best performing approach. Note that P@k is known to be unstable, especially at small values of k [2, 13]. The best performing baseline is the approach of Konstantinovskiy et al. [14], the only other approach using neural embeddings. This points out the effectiveness of neural word embeddings for this task.

The difference in performance between ClaimBuster and the ClaimBuster API is due solely to the quality of the training data (it is otherwise the same approach) and illustrates the effect of training data quality upon model performance. The fact that our model still notably outperforms the ClaimBuster API baseline shows the benefit of an end-to-end learned representation as opposed to a feature engineered one, for this task.

Only ClaimBuster, the approach of Zou et al. [26], and our model obtain notable improvements from the weakly labelled data (ClaimBuster yields a performance similar to that of the ClaimBuster API). TATHYA [19], and the approaches by Gencheva et al. [7] and Konstantinovskiy et al. [14] do not benefit from weak supervision, most likely because feature-engineering is used as opposed to learning the representation.

5.1 Syntactic dependency similarity between check-worthy sentences

Our syntactic dependencies representations aim to discriminate between top weighted words that are common in check-worthy and non-check-worthy sentences based on the syntactic roles of these words (see Section 3.1). To verify this we compute the average overlap of unique syntactic dependency tags between the following three types of randomly sampled sentence pairs: 1) Pairs of n sampled check-worthy sentences, 2) pairs of n sampled non-check-worthy sentences, and 3) mixed pairs of n sampled check-worthy and n sampled non-check-worthy-sentences. We set $n = 10$ and repeat the computations 1000 times. Table 3 displays the resulting average overlaps and their standard deviation. We observe that check-worthy sentence pairs have the highest average overlap of syntactic dependencies, and non-check-worthy the lowest, but both have a similar standard deviation. This indicates that syntactic dependencies are more similar in check-worthy sentences than in non-check-worthy sentences, and as such constitute a good discriminator between check-worthy and non-check-worthy sentences that otherwise contain an overlap of common top-weighted terms. Note that the average overlap of 7 common syntactic dependencies in check-worthy sentences practically applies to approximately half of the words in those sentences (the average sentence length is 14.04 in that dataset – see Table 1). Mixed sentences (both check-worthy and non-check worthy) have an average overlap in between that of check-worthy and non-check-worthy sentences, but with a larger standard deviation, indicating that syntactic dependencies from this mixed group act as a mixed and less stable discriminating signal.

As an example of the overlap problem of common top-weighted terms, consider the check-worthy sentence "*Since president Obama came into office another two million hispanic americans have fallen into poverty*" and non-check-worthy sentence "*I'm running to be a president for all americans*". In these cases the words *president* and *americans* are both important to describe the content, but have different syntactic dependencies (`compound`/`attr` and `nsubj`/`pobj`, respectively).

5.2 Impact of pretrained word embeddings

Our model is the only approach in Table 2 to have word embeddings trained on a domain specific dataset. All other approaches either use no word embeddings (ClaimBuster [8] and TATHYA [19]), use global word embedding averages (Zou et al. [26] and Gencheva et al. [7]), or use a universally trained sentence representation based on global embeddings (Konstantinovskiy et al. [14]). To isolate the effect of these domain-specific trained embeddings upon the performance of our model, we run our method as described in Section 4.3 but vary the pretraining of the embeddings as follows: 1) using no embeddings at all; 2) using randomly initialized embeddings which are not pretrained; 3) using general purpose embeddings pretrained with word2vec on Google News[7]; 4) using our pretrained domain specific embeddings as described in Section 4.2. Table 4 shows the results when varying the embedding strategy. We see that domain specific embeddings (Politics) obtains large improvements – compared to the general purpose embedding – with improvements up to +12-34%. The last row of Table 4 shows the performance without the syntactic dependency parsing (i.e., only the word embedding), which highlights the large performance increase from the syntactic dependency parsing. As expected, no pretraining of the embeddings leads to much lower performance, though MAP is still comparable to most baselines, except for Konstantinovskiy et al. [14]. Not using embeddings at all severely drops overall effectiveness. Collectively these findings show that performance benefits more from training embeddings on smaller, yet domain-specific, data than on much larger but general domain data.

[5] The syntactic dependency parser is available at https://spacy.io/

[7] https://code.google.com/archive/p/word2vec/

	Without Weak Supervision					With Weak Supervision				
	MAP	P@5	P@10	P@20	P@R	MAP	P@5	P@10	P@20	P@R
ClaimBuster API[6]	0.230	0.219	0.176	0.159	0.138	-	-	-	-	-
TATHYA [19]	0.136	0.072	0.062	0.074	0.039	0.147	0.061	0.047	0.060	0.043
ClaimBuster [9]	0.176	0.170	0.112	0.105	0.078	0.224	0.198	0.147	0.138	0.121
Zou et al. [26]	0.187	0.143	0.105	0.099	0.086	0.212	0.171	0.111	0.121	0.097
Gencheva et al. [7]	0.238	0.276	0.170	0.153	0.123	0.236	0.222	0.143	0.138	0.113
Konstantinovskiy et al. [14]	0.267	**0.314**	0.186	0.178	0.149	0.233	0.220	0.146	0.142	0.113
Our model	**0.278**	0.291	**0.194**	**0.193**	**0.159**	**0.302**▲△	**0.344**▲	**0.238**▲△	**0.218**▲△	**0.189**▲△

Table 2: Effectiveness of sentence check-worthiness ranking without and with weak supervision. ▲ marks statistically significant improvements with respect to the overall best baseline at the 0.05 level using a paired two tailed t-test. △ marks statistically significant improvements with respect to the best overall approach without weak supervision at the 0.05 level using a paired two tailed t-test. Significance comparisons are done on the average metric-wise performance in each of the 5 repeated runs.

Sentence pairs	Average Overlap	Standard deviation
Check-worthy	7.00	1.03
Non-check-worthy	4.74	1.08
Mixed	5.64	2.87

Table 3: Average overlap of syntactic dependency tags and its standard deviation between three types of sentence pairs.

Emb. pretrain	MAP	P@5	P@10	P@20	P@R
No embed.	0.184	0.153	0.116	0.103	0.087
No pretraining	0.237	0.230	0.156	0.148	0.130
Google News	0.268	0.262	0.178	0.184	0.143
Politics	**0.302**	**0.344**	**0.238**	**0.218**	**0.189**
Politics w/o syn. dep.	0.285	0.290	0.209	0.202	0.167

Table 4: Performance per type of embedding pretraining. The last row shows the performance without the syntactic dependency parsing.

5.3 Effect of varying the amount of weakly labelled data

We analyse how our model, when used with weak supervision, scales with the amount of weakly labelled data, by reporting performance across the range of 0% to 100% weakly labelled data with 10% increments. At each step we repeat the experiment 5 times with randomly sampled weakly labelled data, and report the average score. Figure 3 displays performance as a function of the percentage of weakly labelled data. As expected, the scores of all metrics generally increase as the amount of data increases. However, the largest increases happen in the first 50% of the data, and then small increases or stagnation for the remaining range up to around 90%. The performance drop at 40% may be explained by the limited number of repetitions of the sampling process, which was done due to runtime considerations. We expect that a larger number of repetitions would smooth out this slight drop.

5.4 Model interpretability

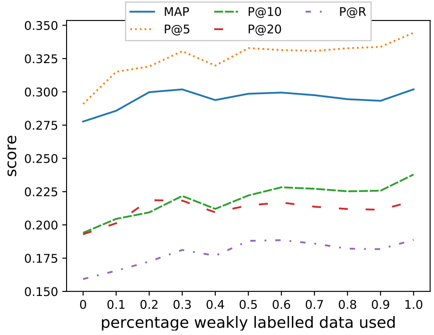


Figure 3: Impact of the amount of weakly labelled data upon our model (0 corresponds to no weakly labelled data).

Check-worthiness detection can be part of semi-automatic or even fully manual fact checking processes, to filter claims that human fact checkers should investigate. In such cases, the output of check-worthiness detection should be easily interpretable by humans. Our model, despite being a deep learning model (generally considered to have low interpretability [24]) – provides easily interpretable output to humans through the attention mechanism that is computed on a word level (see Section 3.2). This score can be used to highlight which parts of a sentence are important for the prediction of check-worthiness. Table 5 illustrates this with a sample of true and false predictions made by our model. We see that sentences with high predicted scores (both correctly and incorrectly predicted as check-worthy) contain a quantifiable fact consisting of a relative large number, e.g. a large amount of money (*trillion dollars, $800 billion*), a high percentage (*60 percent*), or a large collection of entities (*nearly all other presidents combined*). Sentences with low predicted check-worthiness are more varied, but generally either lack a quantifiable element or are generally vague (*buy American and hire American, fix the system, no patience for injustice*). We can also use the model to find seemingly incorrectly labeled sentences, as e.g. the non-check-worthy labelled sentences with high predictions could indeed be labelled as check-worthy instead, e.g. *"our trade deficit in goods with world last year was nearly $800 billion dollars"*.

6 CONCLUSION

We have presented an end-to-end trainable neural network model for ranking check-worthy sentences. The model is pretrained via weak supervision from a large collection of unlabelled data and employs a recurrent neural network with a double representation of each word using domain specific word embeddings and the syntactic dependency parsing of a sentence. We evaluate our model on check-worthy annotated political speeches from the U.S. 2016 presidential election (following the same setting as in the official CLEF 2018 competition on check-worthiness detection [17] but using even more data). Our model does not make use of specialized

Y	$\tilde{Y}$	Sentence
1	0.96	america has spent approximately six trillion dollars in the middle east , all this while our infrastructure at home is crumbling .
1	0.95	today , our total business tax rate is 60 percent higher than our average foreign competitor in the developed world .
1	0.26	its the same reason why she wo nt take responsibility for her central role in unleashing isis all over the world .
1	0.22	we will follow two simple rules ; buy american and hire american .
0	0.04	millions of democrats will join our movement , because we are going to fix the system so it works fairly and justly for all americans .
0	0.05	i have no patience for injustice .
0	0.94	in the last eight years , the past administration has put on more new debt than nearly all of the other presidents combined .
0	0.95	our trade deficit in goods with the world last year was nearly $ 800 billion dollars .

Table 5: Check-worthy and non-check-worthy samples with high and low predictions ($\tilde{Y}$) and ground truth labels (Y). Words are colored according to attention weight: the deeper the shade of red, the larger the attention score assigned.

hand-crafted features as most related work [7, 8, 10, 19], but instead adapts the model and its representation to the domain by being trained in an end-to-end fashion. Thus, our model should by design be able to adapt to other check-worthiness tasks with results depending on the type of discourse and rhetoric. Our model effectively incorporates weak supervision: using an existing check-worthiness ranking system to weakly label political speeches significantly improved performance. Overall, our model outperforms all state of the art baselines in mean average precision and precision at different cut offs, with statistically significant +9-28% gains from the best performing baseline. Further analysis revealed the significance of domain specific word embeddings, compared to traditional general purpose embeddings, and how check-worthy sentences share a syntactic similar structure than non-check-worthy sentences.

Future work consists of investigating further multiple weak signals and incorporating text discourse context into the model.

ACKNOWLEDGMENTS

Partly funded by Innovationsfonden DK, DABAI (5153-00004A), and AMAOS (7076-00121B).

REFERENCES

[1] Mouhamadou Lamine Ba, Laure Berti-Equille, Kushal Shah, and Hossam M Hammady. 2016. Vera: A platform for veracity estimation over web data. In *International Conference Companion on World Wide Web.* 159–162.

[2] Chris Buckley and Ellen M. Voorhees. 2000. Evaluating Evaluation Measure Stability. In *International ACM SIGIR Conference on Research and Development in Information Retrieval.* 33–40.

[3] Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the Properties of Neural Machine Translation: Encoder–Decoder Approaches. *Syntax, Semantics and Structure in Statistical Translation* (2014), 103–111.

[4] Jinho D Choi, Joel Tetreault, and Amanda Stent. 2015. It depends: Dependency parser comparison using a web-based evaluation tool. In *Annual Meeting of the Association for Computational Linguistics.* 387–396.

[5] Giovanni Luca Ciampaglia, Prashant Shiralkar, Luis M Rocha, Johan Bollen, Filippo Menczer, and Alessandro Flammini. 2015. Computational fact checking from knowledge networks. *PloS one* 10, 6 (2015).

[6] Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised Learning of Universal Sentence Representations from Natural Language Inference Data. In *Conference on Empirical Methods in Natural Language Processing.* 670–680.

[7] Pepa Gencheva, Preslav Nakov, Lluís Màrquez, Alberto Barrón-Cedeño, and Ivan Koychev. 2017. A Context-Aware Approach for Detecting Worth-Checking Claims in Political Debates. In *International Conference Recent Advances in Natural Language Processing.* 267–276.

[8] Naeemul Hassan, Fatma Arslan, Chengkai Li, and Mark Tremayne. 2017. Toward automated fact-checking: Detecting check-worthy factual claims by ClaimBuster. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.* 1803–1812.

[9] Naeemul Hassan, Gensheng Zhang, Fatma Arslan, Josue Caraballo, et al. 2017. ClaimBuster: the first-ever end-to-end fact-checking system. *Proceedings of the VLDB Endowment* 10, 12 (2017), 1945–1948.

[10] Israa Jaradat, Pepa Gencheva, Alberto Barrón-Cedeño, Lluís Màrquez, and Preslav Nakov. 2018. ClaimRank: Detecting Check-Worthy Claims in Arabic and English. In *Conference of the North American Chapter of the Association for Computational Linguistics.* 26–30.

[11] Damian Jimenez and Chengkai Li. 2018. An Empirical Study on Identifying Sentences with Salient Factual Statements. In *2018 International Joint Conference on Neural Networks (IJCNN).* IEEE, 1–8.

[12] Georgi Karadzhov, Preslav Nakov, Lluís Màrquez, Alberto Barrón-Cedeño, and Ivan Koychev. 2017. Fully Automated Fact Checking Using External Sources. In *International Conference Recent Advances in Natural Language Processing.* 344–353.

[13] Diane Kelly, Xin Fu, and Chirag Shah. 2010. Effects of position and number of relevant documents retrieved on users' evaluations of system performance. *Transactions on Information Systems (TOIS)* 28, 2 (2010), 9.

[14] Lev Konstantinovskiy, Oliver Price, Mevan Babakar, and Arkaitz Zubiaga. 2018. Towards Automated Factchecking: Developing an Annotation Schema and Benchmark for Consistent Automated Claim Detection. http://arxiv.org/abs/1809.08193

[15] Dieu-Thu Le, Ngoc Thang Vu, and André Blessing. 2016. Towards a text analysis system for political debates. In *LaTeCH@ACL.*

[16] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems.* 3111–3119.

[17] Preslav Nakov, Alberto Barrón-Cedeno, Tamer Elsayed, Reem Suwaileh, et al. 2018. Overview of the CLEF-2018 CheckThat! Lab on automatic identification and verification of political claims. In *International Conference of the CLEF Association.*

[18] Yifan Nie, Alessandro Sordoni, and Jian-Yun Nie. 2018. Multi-level abstraction convolutional model with weak supervision for information retrieval. In *International ACM SIGIR Conference on Research & Development in Information Retrieval.* 985–988.

[19] Ayush Patwari, Dan Goldwasser, and Saurabh Bagchi. 2017. TATHYA: A multi-classifier system for detecting check-worthy statements in political debates. In *ACM on Conference on Information and Knowledge Management.* 2259–2262.

[20] Natalia Silveira, Timothy Dozat, Marie-Catherine de Marneffe, et al. 2014. A Gold Standard Dependency Corpus for English. In *International Conference on Language Resources and Evaluation.* 2897–2904.

[21] James Thorne and Andreas Vlachos. 2018. Automated Fact Checking: Task Formulations, Methods and Future Directions. In *Proceedings of the 27th International Conference on Computational Linguistics.* 3346–3359.

[22] Soroush Vosoughi, Deb Roy, and Sinan Aral. 2018. The spread of true and false news online. *Science* 359, 6380 (2018), 1146–1151.

[23] Hamed Zamani, W. Bruce Croft, and J. Shane Culpepper. 2018. Neural Query Performance Prediction Using Weak Supervision from Multiple Signals. In *International ACM SIGIR Conference on Research and Development in Information Retrieval.* 105–114.

[24] Quanshi Zhang and Song-Chun Zhu. 2018. Visual interpretability for deep learning: a survey. *Frontiers of IT & EE* 19, 1 (2018), 27–39.

[25] Arkaitz Zubiaga, Ahmet Aker, Kalina Bontcheva, Maria Liakata, and Rob Procter. 2018. Detection and resolution of rumours in social media: A survey. *ACM Computing Surveys (CSUR)* 51, 2 (2018), 32.

[26] Chaoyuan Zuo, Ayla Ida Karakas, and Ritwik Banerjee. 2018. A hybrid recognition system for check-worthy claims using heuristics and supervised learning. In *CLEF 2018 Working Notes* (10 ed.). CEUR-WS.org.

Chapter 8

Fact Check-Worthiness Detection with Contrastive Ranking

Casper Hansen, Christian Hansen, Jakob Grue Simonsen, Christina Lioma (2020). Fact Check-Worthiness Detection with Contrastive Ranking. In CLEF, pages 124-130. [37].

Fact Check-Worthiness Detection with Contrastive Ranking

Casper Hansen, Christian Hansen, Jakob Grue Simonsen, and Christina Lioma

Department of Computer Science, University of Copenhagen
{c.hansen,chrh,simonsen,c.lioma}@di.ku.dk

Abstract. Check-worthiness detection aims at predicting which sentences should be prioritized for fact-checking. A typical use is to rank sentences in political debates and speeches according to their degree of check-worthiness. We present the first direct optimization of sentence ranking for check-worthiness; in contrast, all previous work has solely used standard classification based loss functions. We present a recurrent neural network model that learns a sentence encoding, from which a check-worthiness score is predicted. The model is trained by jointly optimizing a binary cross entropy loss, as well as a ranking based pairwise hinge loss. We obtain sentence pairs for training through contrastive sampling, where for each sentence we find the top most semantically similar sentences with opposite label. Through a comparison to existing state-of-the-art check-worthiness methods, we find that our approach improves the MAP score by 11%.

Keywords: check-worthiness · neural networks · contrastive ranking

1 Introduction

Automatic fact-checking systems [10] typically consist of three parts: 1) detect sentences that are interesting to fact-check, 2) gather evidence and background knowledge for each sentence, and 3) manually or automatically estimate veracity. This paper is focused on the first step, where the aim is to detect *check-worthy* sentences for further processing in either a manual or automatic pipeline. The detection can be considered a filtering step in order to limit the computational processing needed in total for the later steps. In practice, sentences are ranked according to their check-worthiness such that they can be processed in order of importance. Thus, the ability to correctly rank check-worthy sentences above non-check-worthy is essential for automatic check-worthiness methods to be useful in practice. However, existing check-worthiness methods [7,3,5] do not directly model this aspect, as they are all based on traditional classification based training objectives.

Motivated by the above, we present a recurrent neural model that learns a sentence encoder for predicting the check-worthiness score of a sentence. Our model is optimizing jointly using a cross entropy classification objective, and–more importantly–also a ranking based objective in the form of a hinge loss.

We construct ranking pairs through contrastive sampling: For each sentence, we find the top k most semantically similar sentences with the opposite label such that the model more accurately learns to identify the (often) subtle differences between normal and check-worthy sentences. Additionally, we use an existing check-worthiness approach to weakly label a large collection of unlabeled political speeches and debates, which is used for pretraining our model [3]. We experimentally evaluate our model on the CLEF-2019 CheckThat! collection of political speeches and debates [1][1], where our approach outperformed the state of the art by 11% on the MAP metric. In a model ablation, we show that both weak supervision and the ranking component improve the results individually (MAP increases of 25% and 9% respectively), while when used together improve the results even more (39% increase).

2 Related Work

Most existing check-worthiness methods are based on feature engineering to extract meaningful features. Given a sentence, ClaimBuster [7] predicts check-worthiness by extracting a set of features (sentiment, statement length, Part-of-Speech (POS) tags, named entities, and tf-idf weighted bag-of-words), and uses a SVM classifier for the prediction. Patwari et al. [9] presented an approach based on similar features, as well as contextual features based on sentences immediately preceding and succeeding the one being assessed, as well as certain hand-crafted POS patterns. The prediction is made by a multi-classifier system based on a dynamic clustering of the data. In the CLEF 2018 competition on check-worthiness detection, Hansen et al. [5] showed that a recurrent neural network with multiple word representations (word embeddings, part-of-speech tagging, and syntactic dependencies) could obtain state-of-the-art results for check-worthiness prediction. Hansen et al. [3] later extended this work with weak supervision based on a large collection of unlabeled political speeches and showed significant improvements compared to existing state-of-the-art methods. This paper directly improves upon [3] by integrating a ranking component into the model trained via contrastive sampling of semantically similar sentences with opposite labels.

3 Neural Check-Worthiness Model

We now present our *Neural Check-Worthiness Model* (*NCWM*), which employs a dual sentence representation, where each word is represented by both a word embedding and its syntactic dependencies within the sentence. The word embedding is a *word2vec* model [8] that aims at capturing the semantics of the sentence. The syntactic dependencies of a word aim to capture the role of that word in modifying the semantics of other words in the sentence. We use a syntactic dependency parser to map each word to its dependency (as a tag) within the sentence, which is then converted to a one-hot encoding. This combination

[1] Our approach ranked 1st in the CLEF-2019 CheckThat! competition [6].

of capturing both semantics and syntactic structure has been shown to work well for predicting check-worthiness [3,5]. For each word in a sentence, the word embedding and one-hot encoding are concatenated and fed to a recurrent neural network with Long Short-Term Memory Units (LSTM) as memory cells:

$$h_i = \mathrm{LSTM}(e_i \oplus o_i) \tag{1}$$

where h_i is the LSTM output for the ith input, e_i is the word embedding of the ith word, o_i is the one-hot syntactic encoding of the ith word, and $\oplus$ is vector concatenation. The output of the LSTM cells are aggregated using an attention weighted sum, where each weight is computed as:

$$\alpha_i = \frac{\exp\left(\mathrm{FF}_{\mathrm{lin}}\left(h_i\right)\right)}{\sum_j \exp\left(\mathrm{FF}_{\mathrm{lin}}\left(h_j\right)\right)} \tag{2}$$

where h_t is the output of the LSTM cell at time t, and $\mathrm{FF}_{\mathrm{lin}}$ is a feed forward layer with linear activation returning a learned scalar. The final check-worthiness score is produced by transforming the weighted LSTM outputs:

$$s = \mathrm{FF}_\sigma\big(\sum_i h_i \alpha_i\big) \tag{3}$$

where FF_σ is a feed forward layer with a sigmoid activation, such that the score lies between 0 and 1.

Loss functions. The model is jointly optimized using both a classification and ranking loss function. For the classification loss, we use the standard binary cross entropy loss:

$$\mathrm{CE(y,\ s)} = -y\log(s) - (1-y)\log(1-s) \tag{4}$$

where y is the ground truth binary label of a sentence and s is the check-worthiness score computed above. For the ranking loss, we use a hinge loss based on the computed check-worthiness scores of sentence pairs with opposite labels. To obtain these pairs we use *contrastive* sampling, such that for each sentence we sample the top k most semantically similar sentences with the opposite label, i.e., for check-worthy sentences we sample k non-check-worthy sentences. To estimate the semantic similarity we compute an average word2vec [8] embedding vector of all words in a sentence, and then use the cosine similarity to find the top k most semantically similar sentences with the opposite label. The purpose of the contrastive sampling is to enable the model to better learn the subtle differences between check-worthy and non-check-worthy sentences. Specifically, for the ith sentence with score s_i, we denote the check-worthiness score of a contrastive sample as s^c, such that the ranking loss is:

$$\mathrm{hinge}(y, s, s^c) = \max\left(0, 1 - \mathrm{sign}(y)(s - s^c)\right) \tag{5}$$

where $\mathrm{sign}(y)$ returns 1 for check-worthy sentences and -1 otherwise. The combination of both the classification and ranking loss enables the model to learn accurate classifications while the predicted scores are sensible for ranking.

4 Experimental Evaluation

We evaluate our approach on the CLEF-2019 CheckThat! dataset [1], which consists of 19 training speeches and debates with a total of 16,421 sentences, where 433 are labeled as being check-worthy (i.e., 2.64% positive samples). The testing set consists of 7 speeches and debates with a total of 7079 sentences, where 110 are labeled as check-worthy (i.e., 1.55% positive samples). We evaluate the performance on the dataset using traditional ranking metrics of MAP and P@k for $k = \{1, 5, 20, 50\}$.

4.1 Tuning

We choose the hyper parameters based on a 19-fold cross validation (1 fold for each training speech and debate). In the following, we list the tuned parameters and underline the optimal values: the LSTM has {50, 100, 150, 200} hidden units, a dropout of {0, 0.1, 0.3, 0.5} was applied to the attention weighted sum, and we used a batch size of {40, 80, 120, 160, 200}. For the contrastive sampling we searched $\{1, \underline{5}, 10, 20\}$ as the number of semantically similar sentences with the opposite label to find for each sentence. For the syntactic dependency parsing we use spaCy[2], and TensorFlow for the neural network implementation.

To train a more generalizable model we employ weak supervision [2,4] by using an online check-worthiness approach[3], to weakly label a large collection of unlabeled political speeches and debates for model pretraining. We obtain 271 political speeches and debates by Hillary Clinton and Donald Trump from the American Presidency Project[4]. Following Hansen et al. [3], we create a domain specific word2vec embedding by crawling documents related to *all* U.S. elections available through the American Presidency Project, e.g., press releases, statements, speeches, and public fundraisers, resulting in 15,059 documents.

4.2 Results

Our Neural Check-Worthiness Model (NCWM) outperformed competitive and state-of-the-art baselines [5,3] with a MAP of 0.1660. To investigate the effect of the ranking component and the weak supervision (see Table 1), we also report the results when these are not part of NCWM. The model without the ranking component is similar to the state-of-the-art work by Hansen et al. [3], and the model without either the ranking component or weak supervision is similar to earlier work by Hansen et al. [5]. The results show that the ranking component and weak supervision lead to notable improvements, both individually and when combined. The inclusion of weak supervision leads to the largest individual MAP improvement (25% increase), while the individual improvement of the ranking

[2] `https://spacy.io/`

[3] `https://idir.uta.edu/claimbuster/`

[4] `https://web.archive.org/web/20170606011755/http://www.presidency.ucsb.edu/`

component is smaller (9% increase). We observe that the ranking component's improvement is marginally larger when weak supervision is included (11% increase with weak supervision compared to 9% without), thus showing that even a weakly labeled signal is also beneficial for learning the correct ranking. Combining both the ranking component and weak supervision leads to a MAP increase of 39% compared to a model without either of them, which highlights the benefit of using both for the task of check-worthiness as the combination provides an improvement larger than the individual parts.

Table 1. Test results for the full Neural Check-Worthiness Model (NCWM) and for the model excluding ranking and weak supervision (WS) components.

Test (Speeches and Debates)	MAP	P@1	P@5	P@20	P@50
NCWM	**0.1660**	**0.2857**	**0.2571**	**0.1571**	**0.1229**
NCWM (w/o. ranking) [3]	0.1496	0.1429	0.2000	0.1429	0.1143
NCWM (w/o. WS)	0.1305	0.1429	0.1714	0.1429	0.1200
NCWM (w/o. ranking and w/o. WS) [5]	0.1195	0.1429	0.1429	0.1143	0.1057

Test (Speeches)	MAP	P@1	P@5	P@20	P@50
NCWM	**0.2502**	**0.5000**	**0.3500**	**0.2375**	**0.1800**
NCWM (w/o. ranking) [3]	0.2256	0.2500	0.3000	0.2250	**0.1800**
NCWM (w/o. WS)	0.1937	0.2500	0.3000	0.2000	0.1600
NCWM (w/o. ranking and w/o. WS) [5]	0.1845	0.2500	0.2500	0.1875	0.1450

Test (Debates)	MAP	P@1	P@5	P@20	P@50
NCWM	**0.0538**	**0.0000**	**0.1333**	0.0500	0.0467
NCWM (w/o. ranking) [3]	0.0482	**0.0000**	0.0667	0.0333	0.0267
NCWM (w/o. WS)	0.0462	**0.0000**	0.0000	**0.0667**	**0.0667**
NCWM (w/o. ranking and w/o. WS) [5]	0.0329	**0.0000**	0.0000	0.0167	0.0533

To investigate the performance on speeches and debates individually, we split the test data and report the performance metrics on each of the sets. In both of them we observe a similar trend as for the full dataset, i.e., that both the ranking component and weak supervision lead to improvements individually and when combined. However, the MAP on the debates is significantly lower than for the speeches (0.0538 and 0.2502 respectively). We believe the reason for this difference is related to two issues: i) All speeches are by Donald Trump and 15 out of 19 training speeches and debates have Donald Trump as a participant, thus the model is better trained to predict sentences by Donald Trump. ii) Debates are often more varied in content compared to a single speech, and contain participants who are not well represented in the training data. Issue (i) can be alleviated by obtaining larger quantities and more varied training data, while issue (ii) may simply be due to debates being inherently more difficult to predict. Models better equipped to handle the dynamics of debates could be a possible direction to solve this.

5 Conclusion

We presented a recurrent neural model that directly models the ranking of check-worthy sentences, which no previous work has done. This was done through a hinge loss based on contrastive sampling, where the most semantically similar sentences with opposite labels were sampled for each sentence. Additionally, we utilize weak supervision through an existing check-worthiness method to label a large unlabeled dataset of political speeches and debates. We experimentally verified that both the sentence ranking and weak supervision lead to notable performance MAP improvements (increases of 9% and 25% respectively) compared to a model without either of them, while using both lead to an improvement greater than the individual parts (39% increase). In comparison to state-of-the-art check-worthiness models, we found our approach to perform 11% better on the MAP metric.

References

1. P. Atanasova, P. Nakov, G. Karadzhov, M. Mohtarami, and G. Da San Martino. Overview of the CLEF-2019 CheckThat! Lab on Automatic Identification and Verification of Claims. Task 1: Check-Worthiness. In *CLEF-2019 CheckThat! Lab*, 2019.
2. M. Dehghani, H. Zamani, A. Severyn, J. Kamps, and W. B. Croft. Neural ranking models with weak supervision. In *ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 65–74, 2017.
3. C. Hansen, C. Hansen, S. Alstrup, J. Grue Simonsen, and C. Lioma. Neural check-worthiness ranking with weak supervision: Finding sentences for fact-checking. In *Companion Proceedings of the World Wide Web Conference*, 2019.
4. C. Hansen, C. Hansen, J. G. Simonsen, S. Alstrup, and C. Lioma. Unsupervised neural generative semantic hashing. In *ACM SIGIR Conference on Research and Development in Information Retrieval*, 2019.
5. C. Hansen, C. Hansen, J. G. Simonsen, and C. Lioma. The copenhagen team participation in the check-worthiness task of the competition of automatic identification and verification of claims in political debates of the clef-2018 fact checking lab. In *CLEF-2018 CheckThat! Lab*, 2018.
6. C. Hansen, C. Hansen, J. G. Simonsen, and C. Lioma. Neural weakly supervised fact check-worthiness detection with contrastive sampling-based ranking loss. In *CLEF-2019 CheckThat! Lab*, 2019.
7. N. Hassan, F. Arslan, C. Li, and M. Tremayne. Toward automated fact-checking: Detecting check-worthy factual claims by claimbuster. In *KDD*, pages 1803–1812, 2017.
8. T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *NeurIPS*, pages 3111–3119, 2013.
9. A. Patwari, D. Goldwasser, and S. Bagchi. Tathya: A multi-classifier system for detecting check-worthy statements in political debates. In *CIKM*, pages 2259–2262, 2017.
10. J. Thorne and A. Vlachos. Automated fact checking: Task formulations, methods and future directions. In *ACL*, pages 3346–3359, 2018.

Chapter 9

Neural Speed Reading with Structural-Jump-LSTM

Christian Hansen, Casper Hansen, Stephen Alstrup, Jakob Grue Simonsen, Christina Lioma (2019). Neural Speed Reading with Structural-Jump-LSTM. In ICLR. [40].

NEURAL SPEED READING WITH STRUCTURAL-JUMP-LSTM

Christian Hansen, Casper Hansen, Stephen Alstrup, Jakob Grue Simonsen & Christina Lioma
Department of Computer Science
University of Copenhagen
Denmark, Copenhagen 2100
{chrh,c.hansen,s.alstrup,simonsen,c.lioma}@di.ku.dk

ABSTRACT

Recurrent neural networks (RNNs) can model natural language by sequentially ”reading” input tokens and outputting a distributed representation of each token. Due to the sequential nature of RNNs, inference time is linearly dependent on the input length, and all inputs are read regardless of their importance. Efforts to speed up this inference, known as ”neural speed reading”, either ignore or skim over part of the input. We present Structural-Jump-LSTM: the first neural speed reading model to both skip and jump text during inference. The model consists of a standard LSTM and two agents: one capable of skipping single words when reading, and one capable of exploiting punctuation structure (sub-sentence separators (,:), sentence end symbols (.!?), or end of text markers) to jump ahead after reading a word. A comprehensive experimental evaluation of our model against all five state-of-the-art neural reading models shows that Structural-Jump-LSTM achieves the best overall floating point operations (FLOP) reduction (hence is faster), while keeping the same accuracy or even improving it compared to a vanilla LSTM that reads the whole text.

1 INTRODUCTION

Recurrent neural networks (RNNs) are a popular model for processing sequential data. The Gated Recurrent Unit (GRU) (Chung et al., 2014) and Long Short Term Memory (LSTM) (Hochreiter & Schmidhuber, 1997) are RNN units developed for learning long term dependencies by reducing the problem of vanishing gradients during training. However, both GRU and LSTM incur fairly expensive computational costs, with e.g. LSTM requiring the computation of 4 fully connected layers for each input it reads, independently of the input's importance for the overall task.

Based on the idea that not all inputs are equally important, and that relevant information can be spread throughout the input sequence, attention mechanisms were developed (Bahdanau et al., 2015) to help the network focus on important parts of the input. With *soft* attention, all inputs are read, but the attention mechanism is fully differentiable. In comparison, *hard* attention completely ignores part of the input sequence. Hard attention mechanisms have been considered in many areas, ranging from computer vision (Mnih et al., 2014; Campos et al., 2018) where the model learns what parts of the image it should focus on, to natural language processing (NLP), such as text classification and question answering (Yu et al., 2017; Campos et al., 2018; Yu et al., 2018), where the model learns which part of a document it can ignore. With hard attention, the RNN has fewer state updates, and therefore fewer floating point operations (FLOPs) are needed for inference. This is often denoted as *speed reading*: obtaining the same accuracy while using (far) fewer FLOPs (Yu et al., 2017; 2018; Seo et al., 2018; Huang et al., 2017; Fu & Ma, 2018). Prior work on speed reading processes text as chunks of either individual words or blocks of contiguous words. If the chunk being read is important enough, a full state update is performed; if not, the chunk is either ignored or a very limited amount of computations are done. This is followed by an action aiming to speed up the reading, e.g. skipping or jumping forward in text.

Inspired by human speed reading, we **contribute** an RNN speed reading model that ignores unimportant words in important sections, while also being able to jump past unimportant sections of the text. Our model, called Structural-Jump-LSTM[1], **both** skips and jumps over dynamically defined chunks of text as follows: (a) it can skip individual words, after reading them, but before updating the RNN state; (b) it uses the punctuation structure of the text to define dynamically spaced jumps to the next sub-sentence separator (,;), end of sentence symbol (.!?), or the end of the text.

An extensive experimental evaluation against *all* state-of-the-art speed reading models (Seo et al., 2018; Yu et al., 2017; 2018; Fu & Ma, 2018; Huang et al., 2017), shows that our Structural-Jump-LSTM of dynamically spaced jumps and word level skipping leads to large FLOP reductions while maintaining the same or better reading accuracy than a vanilla LSTM that reads the full text.

2 Related Work

Prior work on speed reading can be broadly clustered into two groups: 1) jumping based models, and 2) word level skip and skim models, which we outline below.

Jump based models. The method of Yu et al. (2017) reads a fixed number of words, and then may decide to jump a varying number of words ahead (bounded by a maximum allowed amount) in the text, or to jump directly to the end. The model uses a fixed number of total allowed jumps, and the task for the network is therefore to learn how best to spend this *jump budget*. The decision is trained using reinforcement learning, with the REINFORCE algorithm (Williams, 1992), where the reward is defined based only on if the model predicts correctly or not. Thus, the reward does not reflect how much the model has read. The model of Fu & Ma (2018) also has a fixed number of total jumps and is very similar to the work by Yu et al. (2017), however it allows the model to jump both back and forth a varying number of words in order to allow for re-reading important parts. Yu et al. (2018) use a CNN-RNN network where a block of words is first read by a CNN and then read as a single input by the RNN. After each block is read, the network decides to either re-read the block, jump a varying number of blocks ahead, or jump to the end. The decision is trained using reinforcement learning, where both REINFORCE and actor-critic methods were tested, with the actor-critic method leading to more stable training. The reward is based on the loss for the prediction and the FLOPs used by the network to make the prediction. FLOP reduction is thereby directly tied into the reward signal. Huang et al. (2017) propose a simple early-stopping method that uses a RNN and reads on a word level, and where the network learns when to stop. This can be considered a single large jump to the end of the text.

Skip and skim based models. Seo et al. (2018) present a model with two RNNs, a "small" RNN and a "big" RNN. At each time step the model chooses either the big or the small RNN to update the state, based on the input and previous state, which can be considered as text skimming when the small RNN is chosen. This network uses a Gumbel softmax to handle the non-differentiable choice, instead of the more common REINFORCE algorithm. Campos et al. (2018) train a LSTM that may choose to ignore a state update, based on the input. This can be considered as completely skipping a word, and is in contrast to skimming a word as done by Seo et al. (2018). This network uses a straight-through estimator to handle the non-differentiable action choice. This approach is applied on image classification, but we include it in this overview for completeness.

Other speed reading models. Johansen & Socher (2017) introduce a speed reading model for sentiment classification where a simple model with low computational cost first determines if an LSTM model should be used, or a bag-of-words approach is sufficient. The method of Choi et al. (2017) performs question answering by first using a CNN-based sentence classifier to find candidate sentences, thereby making a summary of the whole document relevant for the given query, and then using the summary in an RNN.

More widely, gated units, such as GRU and LSTM, face problems with long input sequences (Neil et al., 2016). Speed reading is one way of handling this problem by reducing the input sequence. In contrast, Neil et al. (2016) handle the problem by only allowing updates to part of the LSTM at a current time point, where an oscillating function controls what part of the LSTM state can currently be updated. Cheng et al. (2016) handle this by using memory networks to store an array of states, and the state at a given point in time comes from applying an attention mechanism over the stored states, handling the issues of older states being written over.

[1] `https://github.com/Varyn/Neural-Speed-Reading-with-Structural-Jump-LSTM`

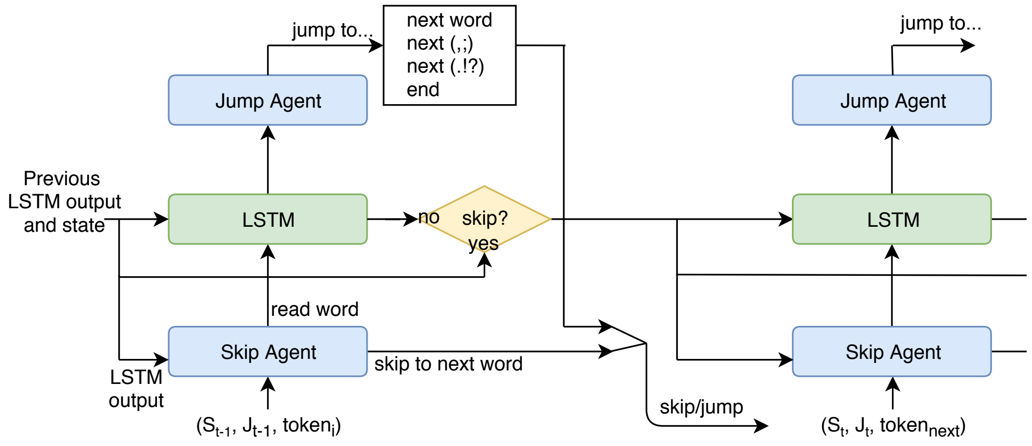


Figure 1: Overview of our proposed model. The input at a given time is the action of the previous skip agent (S_{t-1}), the previous jump agent action (J_{t-1}), and the word embedded token (token_i). token_next corresponds to the next word considered after skipping or jumping. Depending on the skip decision, the no/yes in the diamond shaped skip-box corresponds to which LSTM output and state should be used for the next input (updated or previous respectively).

Overall, the above state-of-the-art models are either jump or skip/skim based. We present the first speed reading model that both jumps and skips. Furthermore, current jumping-based models use a variable jump size for each input, without considering the inherent structure of the text. In contrast, our model defines jumps based on the punctuation structure of the text. This combined approach of both skipping and jumping according to text structure yields notable gains in efficiency (reduced FLOPs) without loss of effectiveness (accuracy). We next present our model.

3 Structural-Jump-LSTM Model

Our Structural-Jump-LSTM model consists of an ordinary RNN with LSTM units and two agents: the *skip* agent and the *jump* agent. Each of these agents compute a corresponding action distribution, where the skip and jump actions are sampled from. The skip agent chooses to either skip a single word, thereby not updating the LSTM, or let the LSTM read the word leading to an update of the LSTM. The jump agent is responsible for jumping forward in the text based on punctuation structure (henceforth referred to as *structural* jumps). A structural jump is a jump to the next word, or the next sub-sentence separator symbol (,;), or the next end of sentence symbol (.!?), or to the end of the text (which is also an instance of end of sentence). The purpose of using two agents is that the skip agent can ignore unimportant words with very little computational overhead, while the jump agent can jump past an unimportant part of the text. As both the skip and jump agent contribute to a reduction in FLOPs (by avoiding LSTM state updates), the Structural-Jump-LSTM is faster at inference than a vanilla LSTM.

Figure 1 shows an overview of our model: The input in each time step is the previous actions of the skip agent (S), of the jump agent (J), and of the current input. The output from the previous LSTM is used in combination with the input to make a *skip decision* – if the word is skipped, the last LSTM state will not be changed. From this we use a standard LSTM cell where the output is fed to the jump agent, and a *jump decision* is made. Both agents make their choice using a fully connected layer, with a size that is significantly smaller than the LSTM cell size, to reduce the number of FLOPs by making the overhead of the agents as small as possible.

Section 3.1 details how inference is done in this model, and Section 3.2 presents how the network is trained.

3.1 Inference

At a given time step t, Structural-Jump-LSTM reads input $x_i \in \mathbb{R}^d$, and the LSTM has a previous output $o_{t-1} \in \mathbb{R}^m$ and state $s_{t-1} \in \mathbb{R}^m$. At time step $t-1$ the skip agent first takes action a_{t-1}^{skip} sampled from the skip-action distribution p_{t-1}^{skip} and the jump agent takes action a_{t-1}^{jump} sampled from the jump-action distribution p_{t-1}^{jump}. If a_{t-1}^{skip} is to skip the word, then the jump agent takes no action, i.e. no jump is made. At time step t the network first needs to sample a_t^{skip} from p_t^{skip}, which is computed in each time step as:

$$p_t^{\text{skip}} = \text{softmax}(d_{\text{LIN}}(\text{state}_t^{\text{skip}})) \tag{1}$$

$$\text{state}_t^{\text{skip}} = d_{\text{ReLU}}(x_t \odot o_{t-1} \odot \text{onehot}(a_{t-1}^{\text{skip}}) \odot \text{onehot}(a_{t-1}^{\text{jump}})) \tag{2}$$

where $d_{activation}$ is a fully connected layer with the given activation, ReLU is the Rectified Linear Unit, LIN is the linear activation, and $\odot$ denotes concatenation of vectors. At inference time the action can either be sampled from the distribution, or chosen greedily, by always choosing the most probable action.

If the action a_t^{skip} = 0, we skip the word and set $o_t = o_{t-1}$ and $s_t = s_{t-1}$, and the network will move to the next word at position $i+1$. If a_t^{skip} = 1, the word is read via the LSTM. The output and new state of the RNN is calculated and produces o_t and s_t for the next step. The probability distribution p_t^{jump} from which action a_t^{jump} is sampled from is computed as:

$$p_t^{\text{jump}} = \text{softmax}(d_{\text{LIN}}(\text{state}_t^{\text{jump}})) \tag{3}$$

$$\text{state}_t^{\text{jump}} = d_{\text{ReLU}}(o_t) \tag{4}$$

If the sampled action a_t^{jump} corresponds to e.g. a jump to the next sentence, then the current LSTM output and state will be kept, and all following inputs will be ignored until a new sentence begins. When there are no more inputs the output of the RNN is used to make a final prediction. If the action is to jump to the end of the text, then the final prediction will be made immediately based on the current output.

3.2 Training

During training, Structural-Jump-LSTM is optimized with regards to two different objectives: 1) Producing an output that can be used for classification, and 2) learning when to skip and jump based on the inputs and their context, such that the minimum number of read inputs gives the maximum accuracy.

For objective 1, the output of the RNN can be used for classification, and the loss is computed as the cross entropy L_{class} against the target.

For objective 2, the agents are non-differentiable due to the discrete sampling of the actions. Thus we choose to reformulate the problem as a reinforcement learning problem, where we define a reward function to maximize. In essence, the reward is given based on the amount read and whether or not the prediction is correct. We denote R as the total reward associated with a sampled sequence of actions, e.g. $a_1^{\text{skip}}, a_2^{\text{skip}} a_2^{\text{jump}}, ..., a_T^{\text{skip}}, a_T^{\text{jump}}$, and R_t as the sum of reward from time t. Note that if the network chooses to skip a word, the sequence will not have a jump at that time step. We use an advantage actor-critic approach (Konda & Tsitsiklis, 2000) to train the agents in order to reduce the variance. The loss for the skip agent is given as:

$$L_{\text{actor}} = -\sum_{t=0}^{T} \log(p_t^{\text{skip}}(a_t^{\text{skip}}|\text{state}_t^{\text{skip}})) \cdot (R_t - V_t^{\text{skip}}) \tag{5}$$

$$V_t^{\text{skip}} = d_{\text{LIN}}(\text{state}_t^{\text{skip}}) \tag{6}$$

where V_t^{skip} is a value estimate of the given state, which is produced by a fully connected layer with output size 1. For the jump agent we do exactly the same as the skip agent, and the sum of the two actor losses is denoted L_{actors}. The value estimate of a state corresponds to how much reward is

dataset	task type	label type	#train	#val	#test	avg. length	vocab. size
IMDB (Maas et al., 2011)	Sentiment	Pos/Neg	21,250	3,750	25,000	230	204,758
Yelp (Zhang et al., 2015)	Sentiment	Pos/Neg	475,999	84,000	37,999	136	644,653
SST (Socher et al., 2013)	Sentiment	Pos/Neg	6,920	872	1,821	19	17,851
Rotten Tomatoes (Pang & Lee, 2005)	Sentiment	Pos/Neg	9,594	1,068	1,068	21	20,388
DBPedia (Lehmann et al., 2015)	Topic	14 topics	475,999	84,000	69,999	47	840,843
AG news (Zhang et al., 2015)	Topic	4 topics	101,999	18,000	7,599	8	41,903
CBT-CN (Hill et al., 2016)	Q/A	10 answers	120,769	2,000	2,500	429	51,774
CBT-NE (Hill et al., 2016)	Q/A	10 answers	108,719	2,000	2,500	394	51,672

Table 1: Dataset statistics.

collected when acting from this state, such that the estimated value is a smoothed function of how much reward the network expects to collect later. Using advantage instead of reward is beneficial as the sign of the loss then depends on whether the achieved reward is higher or lower than the expected reward in a given state. This loss for the agent corresponds to the loss in the popular A3C algorithm (Mnih et al., 2016) in the case of t_{max} being large enough to always reach a terminal condition. This is not a problem in our setting as documents are of finite length. The value estimate is trained together with both agents using the squared difference with the targets being the observed values for each state, (we denote this L_{critics}). Lastly, to provoke exploration in the network we add an entropy loss, where both agents' distributions are targeting a uniform distribution over the actions, (we denote this loss $L_{\text{entropies}}$). The total loss for the network is then:

$$L_{\text{total}} = \alpha L_{\text{class}} + \beta L_{\text{actors}} + \gamma L_{\text{critics}} + \delta L_{\text{entropies}} \tag{7}$$

where $\alpha, \beta, \gamma,$ and δ control the trade-offs between the components.

For each action a reward is given; the reward for a skip action at time t is:

$$r_t^{\text{skip}} = \begin{cases} -\frac{1}{|\text{doc}|} & \text{if } a_t^{\text{skip}} \text{ is a read action} \\ -\frac{c_{\text{skip}}}{|\text{doc}|} & \text{if } a_t^{\text{skip}} \text{ is a skip action} \end{cases} \tag{8}$$

where $|\text{doc}|$ is the number of words in the document such that the reward for skipping a word scales with the document length. The reward is negative for both cases as there is a cost associated with reading a word. The jump action gives no reward, as the reward is implicit by the network collecting less negative reward. At the end an additional reward is given based on whether the network makes a correct prediction, such that the summed reward from time t is given by:

$$R_t = \begin{cases} 1 + w_{\text{rolling}} \sum_{t'=t}^{T} r_{t'}^{\text{skip}} & \text{if } y_{\text{pred}} = y_{\text{target}} \\ p(y_{\text{target}}) + w_{\text{rolling}} \sum_{t'=t}^{T} r_{t'}^{\text{skip}} & \text{if } y_{\text{pred}} \neq y_{\text{true}} \end{cases} \tag{9}$$

y_{pred} is the prediction by the network, y_{true} is the target, and $p(y_{\text{target}})$ is the probability the network gives for the target class. w_{rolling} controls the trade-off between the rolling reward and the reward based on model performance. The final reward is designed such that a large reward is given in the case of a correct prediction, while the agents are still rewarded for increasing the probability for the correct class, even if they did not predict correctly.

4 Experimental Evaluation

We present the experimental evaluation of our method.

4.1 Experimental setup and training

We use the same tasks and datasets used by the state-of-the-art in speed reading (displayed in Table 1), and evaluate against all 5 state-of-the-art models (Seo et al., 2018; Yu et al., 2017; 2018; Fu & Ma, 2018; Huang et al., 2017) in addition to a vanilla LSTM full reading baseline.

For the sentiment and topic classification datasets we apply a fully connected layer on the LSTM output followed by a traditional softmax prediction, where the fully connected layer has the same size as the cell size. On the Question Answering datasets we follow Yu et al. (2017) by choosing the candidate answer with the index that maximizes $\text{softmax}(CWo) \in \mathbb{R}^{10}$, where $C \in \mathbb{R}^{10\times d}$ is the word embedding matrix of the candidate answers, d is the embedding size, $W \in \mathbb{R}^{d\times \text{cell\_size}}$ is a trained weight matrix, and o is the output state of the LSTM. This transforms the answering task into a classification problem with 10 classes. In addition, we read the query followed by the document, to condition reading of the document by the query as done by Yu et al. (2017). On all datasets we initialize the word embedding with GloVe embeddings (Pennington et al., 2014), and use those as the input to the skip agent and LSTM.

We use the predefined train, validation, and testing splits for IMDB, SST, CBT-CN, and CBT-NE, and use 15% of the training data in the rest as validation. For Rotten Tomatoes there is no predefined split, so we set aside 10% for testing as done by Yu et al. (2017). For training the model we use RMSprop with a learning rate chosen from the set $\{0.001, 0.0005\}$, with optimal of 0.001 on question answering datasets (CBT-CN and CBT-NE) and 0.0005 on the topic and sentiment datasets. We use a batch size of 32 on AG news, Rotten Tomatoes, and SST and a batch size of 100 for the remaining. Similarly to Yu et al. (2017), we employ dropout to reduce overfitting, with 0.1 on the embedding and 0.1 on the output of the LSTM. For RNN we use an LSTM cell with a size of 128, and apply gradient clipping with a tresholded value of 0.1. For both agents, their small fully connected layer is fixed to 25 neurons.

On all datasets we train by first maximizing the full read accuracy on the validation set, and the agents are then activated afterwards. While training we include the entropy loss in the total loss to predispose the speed reading to start with a full read behaviour, where the action distributions are initialized to only reading and never skipping or jumping. While maximizing full read accuracy the word embedding is fixed for the Question Answering datasets and trainable on the rest, while being fixed for all datasets during speed read training. As described in Equation 9, w_{rolling} controls the trade-off between correct prediction and speed reading, and was chosen via cross validation from the set $\{0.05, 0.1, 0.15\}$, where most datasets performed best with 0.1. For simplicity we fix the cost of skipping c_{skip} in Equation 8 to 0.5, such that skipping a word costs half of reading a word, which was done to promote jumping behaviour.

For the speed reading phase the total loss, as seen in Equation 7, is a combination of the prediction loss, actor loss, critic loss, and entropy loss, where the actor loss is scaled by a factor of 10 to make it comparable in size to the other losses. The entropy loss controls the amount of exploration and is chosen via cross validation from the set $\{0.01, 0.05, 0.1, 0.15\}$, where most datasets performed best with 0.1. We also cross validate choosing the actions greedily or via sampling from the action distributions, where for QA sampling was optimal and greedy was optimal for the others. Lastly, all non-QA datasets use uniform action target distributions for increased exploration, however CBT-CB and CBT-CN are trained with distribution with 95% probability mass on the ”read” choice of both agents to lower skipping and jumping exploration, which was necessary to stabilize the training.

4.2 Evaluation metrics

The objective of speed reading consists of two opposing forces, as the accuracy of the model should be maximized while reading as few words as possible. We consider two different ways the model can skip a word: i) One or more words can be jumped over, e.g. as done in our model, LSTM-Jump (Yu et al., 2017), Yu-LSTM (Yu et al., 2018) and Adaptive-LSTM (Huang et al., 2017), where the latter implements a jump as early stopping. ii) A word can be skipped or skimmed, where the model is aware of the word (in contrast to jumping), but chooses to do a very limited amount of computations based on it, e.g. as done as skipping in our model or as skimming in Skim-LSTM (Seo et al., 2018). In order to capture both of these speed reading aspects, we report the percentage of words jumped over, and the total reading percentage (when excluding skipped and jumped words).

We calculate the total FLOPs used by the models as done by Seo et al. (2018) and Yu et al. (2018), reported as a FLOP reduction (FLOP-r) between the full read and speed read model. This is done to avoid runtime dependencies on optimized implementations, hardware setups, and whether the model is evaluated on CPU or GPU.

4.3 Results

We now present the results of our evaluation.

(Class: Mean Of transportation) The ~~Alexander Dennis~~ Enviro200 Dart ~~is~~ a midibus manufactured by ~~Alexander Dennis~~ since 2006 ~~for the British~~ market as ~~the~~ successor ~~of the~~ Dart ~~SLF chassis and Pointer body~~. ~~The~~ Enviro200 ~~Dart is manufactured and marketed in North America by New Flyer as the MiDi~~.

Figure 2: Example of jumping and skipping behaviour of our Structural-Jump-LSTM from DBPedia. Skipped words have a single strike-through while jumps consist of a sequence of strikethroughed words. The words that are jumped over or skipped are considered by the model not important for classifying the means of transportation (even though they include several nouns and name entinites that are generally considered to be important (Lioma & van Rijsbergen, 2008).

4.3.1 Structural-Jump-LSTM versus full reading

Table 2 displays the accuracy, the percentage of text being jumped over, and the total reading percentage (when excluding jumped and skipped words), of our approach versus a full reading baseline. Our approach obtains similar accuracies compared to vanilla LSTM, while in some cases reducing the reading percentage to below 20% (IMDB and DBPedia), and with the worst speed reading resulting in a reading percentage of 68.6% (Yelp). The speed reading behaviour varies across the datasets with no skipping on CBT-CN, CBT-NE, and Yelp, no jumping on Rotten Tomatoes, and a mix of skipping and jumping on the remaining datasets.

In 7 out of 8 datasets Structural-Jump-LSTM improves accuracy and in 1 the accuracy is the same (IMDB). At all times, our model reads significantly less text than the full reading baseline, namely 17.5% - 68.8% of the text. Overall this indicates that the jumps/skips of our model are meaningful (if important text was skipped or jumped over, accuracy would drop significantly). An example of this speed reading behaviour (from DBPedia) can be seen in Figure 2. Generally, the model learns to skip some uninformative words, read those it deems important for predicting the target (Mean of transportation), and once it is certain of the prediction it starts jumping to the end of the sentences. Interestingly, in this setting it does not learn to just jump to the end, but rather to inspect the first two words of the last sentence.

4.3.2 Structural-Jump-LSTM versus state-of-the-art speed reading

Table 3 displays the scores of our approach against all five state-of-the-art speed reading models. We report the values from the original papers, which all report the speed reading result with the highest accuracy. We list the reported FLOP reductions when available. If FLOP reductions are not reported in the original paper, we report the speed increase, which should be considered a lower bound on the FLOP reduction. Note that the state-of-the-art models use different network configurations of the RNN network and training schemes, resulting in different full read and speed read accuracies for the same dataset. To allow a consistent comparison of the *effect* of each speed reading model, we report the accuracy difference between each paper's reported full read (vanilla LSTM) and speed read accuracies.

On all datasets our approach provides either the best or shared best FLOP reduction, except on CBT-CN where LSTM-Jump provides a speed increase of 6.1x (compared to 3.9x for our approach). The second best method with regards to FLOP reduction is Skim-LSTM, and the worst is the Adaptive-LSTM model that implements early stopping when the model is certain of its prediction. Skim-LSTM has an evaluation advantage in this FLOP reduction setting, since the FLOP reduction is directly tied to the difference between the small and large LSTM used by the model, such that an unnecessarily large LSTM will lead to very attractive reductions. Skim-LSTM uses a LSTM size of 200 for Question Answering tasks and a default size of 100 for the rest, whereas the small is tested between 5 to 20. In the case of large skimming percentage, it could be argued that the size of the large LSTM could be reduced without affecting the performance. In contrast, jumping based models are less prone to this evaluation flaw because they cannot carry over information from skipped or jumped words.

Most models perform at least as well as a vanilla LSTM. LSTM-Shuttle provides consistent accuracy improvements, but does so at a noticeable FLOP reduction cost compared to Skim-LSTM and our

Model	IMDB			DBPedia			Yelp			AG news		
	Acc	Jump	Read	Acc	Jump	Read	Acc	Jump	Read	Acc	Jump	Read
vanilla LSTM	**0.882**	0%	100%	0.972	0%	100%	0.955	0%	100%	0.880	0%	100%
Structural-Jump-LSTM (ours)	**0.882**	70.7%	19.7%	**0.985**	68.1%	17.5%	**0.958**	31.2%	68.8%	**0.883**	32.2%	52.0%
Model	SST			Rotten Tomatoes			CBT-CN			CBT-NE		
	Acc	Jump	Read	Acc	Jump	Read	Acc	Jump	Read	Acc	Jump	Read
vanilla LSTM	0.837	0%	100%	0.787	0%	100%	0.515	0%	100%	0.453	0%	100%
Structural-Jump-LSTM (ours)	**0.841**	19.1%	53.9%	**0.790**	0.4%	57.8%	**0.522**	67.4%	32.6%	**0.463**	68.7%	31.3%

Table 2: vanilla LSTM refers to a standard LSTM full reading. The columns show the accuracy (Acc), the percentage of text being jumped over, and the total reading percentage.

Model	IMDB		DBPedia		Yelp		AG news	
	ΔAcc	FLOP-r	ΔAcc	FLOP-r	ΔAcc	FLOP-r	ΔAcc	FLOP-r
Structural-Jump-LSTM (ours)	0.000	**6.3x**	**0.013**	**7.0x**	**0.003**	**1.9x**	0.003	**2.4x**
Skim-LSTM (Seo et al., 2018)	0.001	5.8x	-	-	-	-	0.001	1.4x
LSTM-Jump (Yu et al., 2017)	0.003	1.6x*	-	-	-	-	0.012	1.1x*
Yu-LSTM (Yu et al., 2018)	0.005	3.4x	0.002	2.3x	0.002	1.4x	0.001	1.7x
LSTM-Shuttle (Fu & Ma, 2018)	**0.008**	2.1x*	-	-	-	-	**0.020**	1.3x*
Adaptive-LSTM (Huang et al., 2017)	-	-	-0.016	1.1x*	-	-	-0.012	1.1x*
Model	SST		Rotten Tomatoes		CBT-CN		CBT-NE	
	ΔAcc	FLOP-r	ΔAcc	FLOP-r	ΔAcc	FLOP-r	ΔAcc	FLOP-r
Structural-Jump-LSTM (ours)	**0.004**	**2.4x**	0.003	**2.1x**	0.007	3.9x	0.010	**4.1x**
Skim-LSTM (Seo et al., 2018)	0.000	**2.4x**	**0.017**	**2.1x**	0.014	1.8x	0.024	3.6x
LSTM-Jump (Yu et al., 2017)	-	-	0.002	1.5x*	**0.044**	**6.1x***	**0.030**	3.0x*
Yu-LSTM (Yu et al., 2018)	-	-	-	-	-	-	-	-
LSTM-Shuttle (Fu & Ma, 2018)	-	-	0.007	1.7x*	-	-	0.019	3.0x*
Adaptive-LSTM (Huang et al., 2017)	-	-	-	-	-	-	-	-

Table 3: Comparison of state-of-the-art speed reading models. ΔAcc is the difference between the accuracy of the full read LSTM and the model (the higher the better), and FLOP-r is the FLOP reduction compared to a full read model. A star (*) indicates that the original paper provided only a speed increase, which should be considered a lower bound for FLOP-r.

approach. This can be explained by its ability to make backwards jumps in the text in order to reread important parts, which is similar to the idea of Yu-LSTM. The largest accuracy improvements appear on CBT-CN and CBT-NE with LSTM-Jump. The performance obtained by reading just the query has been reported to be very similar to using both the query and the 20 sentences (Hill et al., 2016), which could indicate a certain noise level in the data that speed reading models are able to identify and reduce the number of read words between the high information section of the text and the final prediction. LSTM-Jump and LSTM-Shuttle are optimized via maximizing a jumping budget, where only a certain specified number of jumps are allowed to be made, which provides an edge in comparison to the other methods in this setting because prior knowledge about the high information level of the query can be encoded in the budget (cf. Yu et al. (2017) where the best accuracy is obtained using 1 jump for CBT-CN and 5 for CBT-NE). In the setting of speed reading the query is read first to condition the jumping based on the query – this makes the model very likely to prefer jumping shortly after the query is read, to not degrade the LSTM state obtained after reading the query. Overall, budgets can be beneficial if prior information about the document is available, but this is most often not the case for a large set of real world datasets. However, methods based on budgets are in general significantly more rigid, as every document in a collection has the same budget, but the required budget for each document is not necessarily the same.

5 Conclusion

We presented Structural-Jump-LSTM, a recurrent neural network for speed reading. Structural-Jump-LSTM is inspired by human speed reading, and can skip irrelevant words in important sections, while also jumping past unimportant parts of a text. It uses the dynamically spaced punctuation structure of text to determine whether to jump to the next word, the next sub-sentence separator (,;), next end of sentence (.!?), or to the end of the text. In addition, it allows skipping a word after observing it without updating the state of the RNN. Through an extensive experimental evaluation against all five state-of-the-art baselines, Structural-Jump-LSTM obtains the overall largest reduction in floating point operations, while maintaining the same accuracy or even improving it over a vanilla LSTM model that reads the full text. We contribute the first ever neural speed reading model that both skips and jumps over dynamically defined chunks of text without loss of effectiveness and with notable gains in efficiency. Future work includes investigating other reward functions, where most of the reward is not awarded in the end, and whether this would improve agent training by having a stronger signal spread throughout the text.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *ICLR*, 2015.

Víctor Campos, Brendan Jou, Xavier Giró-i Nieto, Jordi Torres, and Shih-Fu Chang. Skip rnn: Learning to skip state updates in recurrent neural networks. *ICLR*, 2018.

Jianpeng Cheng, Li Dong, and Mirella Lapata. Long short-term memory-networks for machine reading. *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2016.

Eunsol Choi, Daniel Hewlett, Jakob Uszkoreit, Illia Polosukhin, Alexandre Lacoste, and Jonathan Berant. Coarse-to-fine question answering for long documents. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pp. 209–220, 2017.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

Tsu-Jui Fu and Wei-Yun Ma. Speed reading: Learning to read forbackward via shuttle. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2018.

Felix Hill, Antoine Bordes, Sumit Chopra, and Jason Weston. The goldilocks principle: Reading children's books with explicit memory representations. In *Proceedings of the 4th International Conference on Learning Representations (ICLR 2016)*, San Juan, Puerto Rico, May 2016.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8): 1735–1780, 1997.

Zhengjie Huang, Zi Ye, Shuangyin Li, and Rong Pan. Length adaptive recurrent model for text classification. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pp. 1019–1027. ACM, 2017.

Alexander Johansen and Richard Socher. Learning when to skim and when to read. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*, pp. 257–264, 2017.

Vijay R Konda and John N Tsitsiklis. Actor-critic algorithms. In *Advances in neural information processing systems*, pp. 1008–1014, 2000.

Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick Van Kleef, Sören Auer, et al. Dbpedia–a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web*, 6(2):167–195, 2015.

Christina Lioma and CJ Keith van Rijsbergen. Part of speech n-grams and information retrieval. *French Review of applied linguistics*, 13(1):9–22, 2008.

Andrew L Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies-volume 1*, pp. 142–150. Association for Computational Linguistics, 2011.

Volodymyr Mnih, Nicolas Heess, Alex Graves, et al. Recurrent models of visual attention. In *Advances in neural information processing systems*, pp. 2204–2212, 2014.

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pp. 1928–1937, 2016.

Daniel Neil, Michael Pfeiffer, and Shih-Chii Liu. Phased lstm: Accelerating recurrent network training for long or event-based sequences. In *Advances in Neural Information Processing Systems*, pp. 3882–3890, 2016.

Bo Pang and Lillian Lee. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd annual meeting on association for computational linguistics*, pp. 115–124. Association for Computational Linguistics, 2005.

Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pp. 1532–1543, 2014.

Minjoon Seo, Sewon Min, Ali Farhadi, and Hannaneh Hajishirzi. Neural speed reading via skim-rnn. *ICLR*, 2018.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pp. 1631–1642, 2013.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

Adams Wei Yu, Hongrae Lee, and Quoc Le. Learning to skim text. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 1880–1890, 2017. doi: 10.18653/v1/P17-1172. URL `http://www.aclweb.org/anthology/P17-1172`.

Keyi Yu, Yang Liu, Alexander G. Schwing, and Jian Peng. Fast and accurate text classification: Skimming, rereading and early stopping, 2018. URL `https://openreview.net/forum?id=ryZ8sz-Ab`.

Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, pp. 649–657, 2015.

Bibliography

[1] Antti Ajanki, David R. Hardoon, Samuel Kaski, Kai Puolamäki, and John Shawe-Taylor. Can eyes reveal interest? implicit queries from gaze patterns. *User Model. User-Adapt. Interact.*, 19(4):307–339, 2009.

[2] Liesbeth Allein, Isabelle Augenstein, and Marie-Francine Moens. Time-aware evidence ranking for fact-checking. *arXiv preprint arXiv:2009.06402*, 2020.

[3] Stephen Alstrup, Casper Hansen, Christian Hansen, Niklas Hjuler, Stephan Lorenzen, and Ninh Pham. Dabai: A data driven project for e-learning in denmark. In *ECEL17 - Proceedings of the 16th European Conference on e-Learning*, 2017.

[4] Ashton Anderson, Ravi Kumar, Andrew Tomkins, and Sergei Vassilvitskii. The dynamics of repeat consumption. In *Proceedings of the 23rd international conference on World wide web*, pages 419–430, 2014.

[5] Ashton Anderson, Lucas Maystre, Ian Anderson, Rishabh Mehrotra, and Mounia Lalmas. Algorithmic effects on the diversity of consumption on spotify. In *Proceedings of The Web Conference 2020*, pages 2155–2165, 2020.

[6] Azin Ashkan, Branislav Kveton, Shlomo Berkovsky, and Zheng Wen. Optimal greedy diversity for recommendation. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

[7] Isabelle Augenstein, Christina Lioma, Dongsheng Wang, Lucas Chaves Lima, Casper Hansen, Christian Hansen, and Jakob Grue Simonsen. MultiFC: A real-world multi-domain dataset for evidence-based fact checking of claims. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4685–4697, Hong Kong, China, 2019. Association for Computational Linguistics.

[8] Mouhamadou Lamine Ba, Laure Berti-Equille, Kushal Shah, and Hossam M Hammady. Vera: A platform for veracity estimation over web data. In *Pro-*

ceedings of the 25th international conference companion on world wide web, pages 159–162, 2016.

[9] Yoram Bachrach, Yehuda Finkelstein, Ran Gilad-Bachrach, Liran Katzir, Noam Koenigstein, Nir Nice, and Ulrich Paquet. Speeding up the xbox recommender system using a euclidean transformation for inner-product spaces. In *Proceedings of the 8th ACM Conference on Recommender systems*, pages 257–264, 2014.

[10] Punam Bedi, Shikha Agarwa, Archana Singhal, Ena Jain, and Gunjan Gupta. A novel semantic clustering approach for reasonable diversity in news recommendations. In *Computational Intelligence in Data Mining*. 2015.

[11] Alex Beutel, Paul Covington, Sagar Jain, Can Xu, Jia Li, Vince Gatto, and Ed H Chi. Latent cross: Making use of context in recurrent recommender systems. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, pages 46–54, 2018.

[12] Keith Bradley and Barry Smyth. Improving recommendation diversity. In *Proceedings of the Twelfth Irish Conference on Artificial Intelligence and Cognitive Science, Maynooth, Ireland*, volume 85, pages 141–152, 2001.

[13] Brian Brost, Rishabh Mehrotra, and Tristan Jehan. The music streaming sessions dataset. In *The World Wide Web Conference*, pages 2594–2600, 2019.

[14] Georg Buscher, Andreas Dengel, Ralf Biedert, and Ludger V. Elst. Attentive documents: Eye tracking as implicit feedback for information retrieval and beyond. *ACM Trans. Interact. Intell. Syst.*, 1(2):30, 2012.

[15] Georg Buscher, Andreas Dengel, and Ludger van Elst. Query expansion using gaze-based feedback on the subdocument level. In *SIGIR*, pages 387–394, 2008.

[16] Víctor Campos, Brendan Jou, Xavier Giró-i Nieto, Jordi Torres, and Shih-Fu Chang. Skip rnn: Learning to skip state updates in recurrent neural networks. *ICLR*, 2018.

[17] Toni Cebrián, Marc Planagumà, Paulo Villegas, and Xavier Amatriain. Music recommendations with temporal context awareness. In *Proceedings of the fourth ACM conference on Recommender systems*, pages 349–352, 2010.

[18] Jun Chen, Chaokun Wang, and Jianmin Wang. Will you" reconsume" the near past? fast prediction on short-term reconsumption behaviors. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 29, 2015.

[19] Giovanni Luca Ciampaglia, Prashant Shiralkar, Luis M Rocha, Johan Bollen, Filippo Menczer, and Alessandro Flammini. Computational fact checking from knowledge networks. *PloS one*, 10(6):e0128193, 2015.

[20] Hanjun Dai, Yichen Wang, Rakshit Trivedi, and Le Song. Deep coevolutionary network: Embedding user and item features for recommendation. *arXiv preprint arXiv:1609.03675*, 2016.

[21] Mostafa Dehghani, Hamed Zamani, Aliaksei Severyn, Jaap Kamps, and W Bruce Croft. Neural ranking models with weak supervision. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 65–74, 2017.

[22] Tsu-Jui Fu and Wei-Yun Ma. Speed reading: Learning to read forbackward via shuttle. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2018.

[23] Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. Convolutional sequence to sequence learning. In *International Conference on Machine Learning*, pages 1243–1252, 2017.

[24] Pepa Gencheva, Preslav Nakov, Lluís Màrquez, Alberto Barrón-Cedeño, and Ivan Koychev. A context-aware approach for detecting worth-checking claims in political debates. In *Proceedings of the International Conference Recent Advances in Natural Language Processing, RANLP 2017*, pages 267–276, 2017.

[25] Guibing Guo. Resolving data sparsity and cold start in recommender systems. In *International Conference on User Modeling, Adaptation, and Personalization*, pages 361–364. Springer, 2012.

[26] Jacek Gwizdka. News stories relevance effects on eye-movements. In *ETRA*, pages 283–286, 2014.

[27] Casper Hansen, Christian Hansen, Stephen Alstrup, and Christina Lioma. Smart city analytics: Ensemble-learned prediction of citizen home care. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, CIKM 2017, Singapore, November 06 - 10, 2017*, pages 2095–2098. ACM, 2017.

[28] Casper Hansen, Christian Hansen, Stephen Alstrup, Jakob Grue Simonsen, and Christina Lioma. Contextually propagated term weights for document representation. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2019, Paris, France, July 21-25, 2019*, pages 897–900. ACM, 2019.

[29] Casper Hansen, Christian Hansen, Stephen Alstrup, Jakob Grue Simonsen, and Christina Lioma. Neural check-worthiness ranking with weak supervision: Finding sentences for fact-checking. In *Companion of The 2019 World Wide Web Conference, WWW 2019, San Francisco, CA, USA, May 13-17, 2019*, pages 994–1000. ACM, 2019.

[30] Casper Hansen, Christian Hansen, Stephen Alstrup, Jakob Grue Simonsen, and Christina Lioma. Unsupervised multi-index semantic hashing. In *The 2021 World Wide Web Conference, WWW 2021, Ljubljana, Slovenia, April 19-23, 2021*, page in press. ACM, 2021.

[31] Casper Hansen, Christian Hansen, Lucas Maystre, Rishabh Mehrotra, Brian Brost, Federico Tomasi, and Mounia Lalmas. Contextual and sequential user embeddings for large-scale music recommendation. In *RecSys 2020: Fourteenth ACM Conference on Recommender Systems, Virtual Event, Brazil, September 22-26, 2020*, pages 53–62. ACM, 2020.

[32] Casper Hansen, Christian Hansen, Jakob Grue Simonsen, Stephen Alstrup, and Christina Lioma. Unsupervised neural generative semantic hashing. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2019, Paris, France, July 21-25, 2019*, pages 735–744. ACM, 2019.

[33] Casper Hansen, Christian Hansen, Jakob Grue Simonsen, Stephen Alstrup, and Christina Lioma. Content-aware neural hashing for cold-start recommendation. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, SIGIR 2020, Virtual Event, China, July 25-30, 2020*, pages 971–980. ACM, 2020.

[34] Casper Hansen, Christian Hansen, Jakob Grue Simonsen, Stephen Alstrup, and Christina Lioma. Unsupervised semantic hashing with pairwise reconstruction. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, SIGIR 2020, Virtual Event, China, July 25-30, 2020*, pages 2009–2012. ACM, 2020.

[35] Casper Hansen, Christian Hansen, Jakob Grue Simonsen, and Christina Lioma. The copenhagen team participation in the check-worthiness task of the competition of automatic identification and verification of claims in political debates of the clef-2018 fact checking lab. In *CLEF-2018 CheckThat! Lab*, 2018.

[36] Casper Hansen, Christian Hansen, Jakob Grue Simonsen, and Christina Lioma. Neural weakly supervised fact check-worthiness detection with contrastive sampling-based ranking loss. In *CLEF-2019 CheckThat! Lab*, 2019.

[37] Casper Hansen, Christian Hansen, Jakob Grue Simonsen, and Christina Lioma. Fact check-worthiness detection with contrastive ranking. In *Experimental IR Meets Multilinguality, Multimodality, and Interaction - 11th International Conference of the CLEF Association, CLEF 2020, Thessaloniki, Greece, September 22-25, 2020, Proceedings*, volume 12260 of *Lecture Notes in Computer Science*, pages 124–130. Springer, 2020.

[38] Casper Hansen, Christian Hansen, Jakob Grue Simonsen, and Christina Lioma. Projected hamming dissimilarity for bit-level importance coding in collaborative filtering. In *The 2021 World Wide Web Conference, WWW 2021, Ljubljana, Slovenia, April 19-23, 2021*, page in press. ACM, 2021.

[39] Christian Hansen, Casper Hansen, Stephen Alstrup, and Christina Lioma. Modelling end-of-session actions in educational systems. In *Proceedings of the 12th International Conference on Educational Data Mining, EDM 2019, Montréal, Canada, July 2-5, 2019*. International Educational Data Mining Society (IEDMS), 2019.

[40] Christian Hansen, Casper Hansen, Stephen Alstrup, Jakob Grue Simonsen, and Christina Lioma. Neural speed reading with structural-jump-lstm. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*, 2019.

[41] Christian Hansen, Casper Hansen, Niklas Hjuler, Stephen Alstrup, and Christina Lioma. Sequence modelling for analysing student interaction with educational systems. In *Proceedings of the 10th International Conference on Educational Data Mining, EDM 2017, Wuhan, Hubei, China, June 25-28, 2017*. International Educational Data Mining Society (IEDMS), 2017.

[42] Christian Hansen, Casper Hansen, Jakob Grue Simonsen, Stephen Alstrup, and Christina Lioma. Modelling sequential music track skips using a multi-rnn approach. In *WSDM Cup*. ACM, 2019.

[43] Christian Hansen, Casper Hansen, Jakob Grue Simonsen, Birger Larsen, Stephen Alstrup, and Christina Lioma. Factuality checking in news headlines with eye tracking. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, SIGIR 2020, Virtual Event, China, July 25-30, 2020*, pages 2013–2016. ACM, 2020.

[44] Christian Hansen, Rishabh Mehrotra, Casper Hansen, Brian Brost, Lucas Maystre, and Mounia Lalmas. Shifting consumption towards diverse content on music streaming platforms. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, WSDM '21, page 238–246. Association for Computing Machinery, 2021.

[45] David Hardoon, John Shawe-Taylor, Antti Ajanki, Kai Puolamäki, and Samuel Kaski. Information retrieval by inferring implicit queries from eye movements. *Journal of Machine Learning Research - Proceedings Track*, 2:179–186, 12 2007.

[46] Naeemul Hassan, Fatma Arslan, Chengkai Li, and Mark Tremayne. Toward automated fact-checking: Detecting check-worthy factual claims by claimbuster. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1803–1812, 2017.

[47] Lee Howell et al. Digital wildfires in a hyperconnected world. *WEF report*, 45(3):15–94, 2013.

[48] Israa Jaradat, Pepa Gencheva, Alberto Barrón-Cedeño, Lluís Màrquez, and Preslav Nakov. Claimrank: Detecting check-worthy claims in arabic and english. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*, pages 26–30, 2018.

[49] Espen Jimenez-Solem, Tonny S Petersen, Casper Hansen, Christian Hansen, Christina Lioma, Christian Igel, Wouter Boomsma, Oswin Krause, Stephan Lorenzen, Raghavendra Selvan, et al. Developing and validating covid-19 adverse outcome risk prediction models from a bi-national european cohort of 5594 patients. *Scientific reports*, 11(1):1–12, 2021.

[50] Thorsten Joachims, Adith Swaminathan, and Tobias Schnabel. Unbiased learning-to-rank with biased feedback. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, pages 781–789, 2017.

[51] Marius Kaminskas, Francesco Ricci, and Markus Schedl. Location-aware music recommendation using auto-tagging and hybrid matching. In *Proceedings of the 7th ACM Conference on Recommender Systems*, pages 17–24, 2013.

[52] Georgi Karadzhov, Preslav Nakov, Lluís Màrquez, Alberto Barrón-Cedeño, and Ivan Koychev. Fully automated fact checking using external sources. In *Proceedings of the International Conference Recent Advances in Natural Language Processing, RANLP 2017*, pages 344–353, 2017.

[53] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

[54] Matevž Kunaver and Tomaž Požrl. Diversity in recommender systems–a survey. *Knowledge-based systems*, 123:154–162, 2017.

[55] Neal Lathia, Stephen Hailes, Licia Capra, and Xavier Amatriain. Temporal diversity in recommender systems. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, pages 210–217, 2010.

[56] Dieu-Thu Le, Ngoc Thang Vu, and Andre Blessing. Towards a text analysis system for political debates. In *Proceedings of the 10th SIGHUM Workshop on Language Technology for Cultural Heritage, Social Sciences, and Humanities*, pages 134–139, 2016.

[57] Lucas Chaves Lima, Casper Hansen, Christian Hansen, Dongsheng Wang, Maria Maistro, Birger Larsen, Jakob Grue Simonsen, and Christina Lioma. Denmark's participation in the search engine trec covid-19 challenge: Lessons learned about searching for precise biomedical scientific information on covid-19. In *TREC COVID-19 Challenge*, 2021.

[58] Tomasz D. Loboda, Peter Brusilovsky, and Jörg Brunstein. Inferring word relevance from eye-movements of readers. In *IUI*, pages 175–184, 2011.

[59] Rastin Matin, Casper Hansen, Christian Hansen, and Pia Mølgaard. Predicting distresses using deep learning of text segments in annual reports. *Expert Systems with Applications*, 132:199–208, 2019.

[60] Rada Mihalcea and Carlo Strapparava. The lie detector: Explorations in the automatic recognition of deceptive language. In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, pages 309–312, 2009.

[61] Preslav Nakov, Alberto Barrón-Cedeno, Tamer Elsayed, Reem Suwaileh, Lluís Màrquez, Wajdi Zaghouani, Pepa Atanasova, Spas Kyuchukov, and Giovanni Da San Martino. Overview of the clef-2018 checkthat! lab on automatic identification and verification of political claims. In *International conference of the cross-language evaluation forum for european languages*, pages 372–387. Springer, 2018.

[62] Yifan Nie, Alessandro Sordoni, and Jian-Yun Nie. Multi-level abstraction convolutional model with weak supervision for information retrieval. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 985–988, 2018.

[63] Minsu Park, Jennifer Thom, Sarah Mennicken, Henriette Cramer, and Michael Macy. Global music streaming data reveal diurnal and seasonal patterns of affective preference. *Nature human behaviour*, 3(3):230–236, 2019.

[64] Francesco Sanna Passino, Lucas Maystre, Dmitrii Moor, Ashton Anderson, and Mounia Lalmas. Where to next? a dynamic model of user preferences. In *The 2021 World Wide Web Conference, WWW 2021, Ljubljana, Slovenia, April 19-23, 2021*, page in press. ACM, 2021.

[65] Ayush Patwari, Dan Goldwasser, and Saurabh Bagchi. Tathya: A multi-classifier system for detecting check-worthy statements in political debates. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 2259–2262, 2017.

[66] Michael J Pazzani and Daniel Billsus. Content-based recommendation systems. In *The adaptive web*, pages 325–341. Springer, 2007.

[67] Terry F Pettijohn, Greg M Williams, and Tiffany C Carter. Music for the seasons: seasonal music preferences in college students. *Current psychology*, 29(4):328–345, 2010.

[68] Kashyap Popat, Subhabrata Mukherjee, Andrew Yates, and Gerhard Weikum. Declare: Debunking fake news and false claims using evidence-aware deep learning. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 22–32, 2018.

[69] Kai Puolamäki, Antti Ajanki, and Samuel Kaski. Learning to learn implicit queries from gaze patterns. In *ICML*, pages 760–767, 2008.

[70] Alessandro Raganato, Claudio Delli Bovi, and Roberto Navigli. Neural sequence learning models for word sense disambiguation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1156–1167, Copenhagen, Denmark, September 2017. Association for Computational Linguistics.

[71] Pengjie Ren, Zhumin Chen, Jing Li, Zhaochun Ren, Jun Ma, and Maarten De Rijke. Repeatnet: A repeat aware neural recommendation machine for session-based recommendation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4806–4813, 2019.

[72] Marco Tulio Ribeiro, Nivio Ziviani, Edleno Silva De Moura, Itamar Hata, Anisio Lacerda, and Adriano Veloso. Multiobjective pareto-efficient approaches for recommender systems. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 5(4):53, 2015.

[73] Matthew J Salganik, Peter Sheridan Dodds, and Duncan J Watts. Experimental study of inequality and unpredictability in an artificial cultural market. *science*, 311(5762):854–856, 2006.

[74] Markus Schedl, Hamed Zamani, Ching-Wei Chen, Yashar Deldjoo, and Mehdi Elahi. Current challenges and visions in music recommender systems research. *International Journal of Multimedia Information Retrieval*, 7(2):95–116, 2018.

[75] Minjoon Seo, Sewon Min, Ali Farhadi, and Hannaneh Hajishirzi. Neural speed reading via skim-rnn. *ICLR*, 2018.

[76] Shaden Shaar, Nikolay Babulkov, Giovanni Da San Martino, and Preslav Nakov. That is a known lie: Detecting previously fact-checked claims. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3607–3618, 2020.

[77] Michael Sülflow, Svenja Schäfer, and Stephan Winter. Selective attention in the news feed: An eye-tracking study on the perception and selection of

political news posts on facebook. *New Media & Society*, 21(1):168–190, 2019.

[78] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

[79] James Thorne and Andreas Vlachos. Automated fact checking: Task formulations, methods and future directions. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3346–3359, 2018.

[80] James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. Fever: a large-scale dataset for fact extraction and verification. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 809–819, 2018.

[81] Aäron Van Den Oord, Sander Dieleman, and Benjamin Schrauwen. Deep content-based music recommendation. In *Neural Information Processing Systems Conference (NIPS 2013)*, volume 26. Neural Information Processing Systems Foundation (NIPS), 2013.

[82] Soroush Vosoughi, Deb Roy, and Sinan Aral. The spread of true and false news online. *Science*, 359(6380):1146–1151, 2018.

[83] Dongsheng Wang, Casper Hansen, Lucas Chaves Lima, Christian Hansen, Maria Maistro, Jakob Grue Simonsen, and Christina Lioma. Multi-head self-attention with role-guided masks. In *Proceedings of the 43rd European Conference on Information Retrieval Research*, 2021.

[84] Jacek Wasilewski and Neil Hurley. Incorporating diversity in a learning to rank recommender system. In *The Twenty-Ninth International Flairs Conference*, 2016.

[85] Dustin Wright and Isabelle Augenstein. Claim check-worthiness detection as positive unlabelled learning. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, pages 476–488, 2020.

[86] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016.

[87] Adams Wei Yu, Hongrae Lee, and Quoc Le. Learning to skim text. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1880–1890, 2017.

[88] Keyi Yu, Yang Liu, Alexander G. Schwing, and Jian Peng. Fast and accurate text classification: Skimming, rereading and early stopping, 2018.

[89] Hamed Zamani, W Bruce Croft, and J Shane Culpepper. Neural query performance prediction using weak supervision from multiple signals. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 105–114, 2018.

[90] Yuan Cao Zhang, Diarmuid Ó Séaghdha, Daniele Quercia, and Tamas Jambor. Auralist: introducing serendipity into music recommendation. In *Proceedings of the fifth ACM international conference on Web search and data mining*, pages 13–22, 2012.

[91] Lin Zhu and Yihong Chen. Session-based sequential skip prediction via recurrent neural networks. *arXiv preprint arXiv:1902.04743*, 2019.

[92] Cai-Nicolas Ziegler, Sean M McNee, Joseph A Konstan, and Georg Lausen. Improving recommendation lists through topic diversification. In *Proceedings of the 14th international conference on World Wide Web*, pages 22–32, 2005.

[93] Arkaitz Zubiaga, Ahmet Aker, Kalina Bontcheva, Maria Liakata, and Rob Procter. Detection and resolution of rumours in social media: A survey. *ACM Computing Surveys (CSUR)*, 51(2):1–36, 2018.

[94] Arkaitz Zubiaga, Maria Liakata, Rob Procter, Geraldine Wong Sak Hoi, and Peter Tolmie. Analysing how people orient to and spread rumours in social media by looking at conversational threads. *PloS one*, 11(3):e0150989, 2016.