That ain't how I speak: Personalizing natural language processing

Milton King

Master of Computer Science, University of New Brunsiwck, 2017

A Dissertation Submitted in Partial Fulfilment of the Requirements for the Degree of

Doctor of Philosophy

in the Graduate Academic Unit of Computer Science

Supervisor:	Paul Cook, PhD, Computer Science
Examining Board:	Michael Fleming, PhD, Computer Science Scott Bateman PhD, Computer Science Christine Horne PhD, French
External Examiner:	Ted Pederson, PhD, Computer Science, University of Minnesota Duluth

This dissertation is accepted by the
Dean of Graduate Studies

THE UNIVERSITY OF NEW BRUNSWICK

October, 2021

Abstract

Natural language processing (NLP) involves automatically analyzing text written by human authors. People develop their own use of a language known as an idiolect, which could result in poor performance from generic NLP systems. Ideally, each person would have their own personalized system that is tailored toward them. In this thesis, I demonstrate the potential benefits of personalizing systems in three different NLP tasks, which include language modeling (estimating the probability of a sequence of words), authorship verification (determining if a document belongs to a specific person), and word sense disambiguation (assigning a dictionary-like meaning to a word in context). Personalization in these topics has not been widely studied and to the best of my knowledge, this is the first work to consider personalization with word sense disambiguation, for which I design a novel dataset. For each task, I show the increase in performance that the proposed personalized models have against state-of-the-art models. The experiments in this thesis are designed without consideration of people's demographic and all personalized methods require relatively low amounts of text from an individual. These two criteria are respected to ensure the personalized methods work well for each individual regardless of their demographic or the amount of text they have authored.

Acknowledgements

This work was made possible through the continued support of many individuals. I would like to extend a warm thanks to the following people:

To my parents, who never hesitated to assist me when I needed it.

To my sister and her crew, who were always excited when I visited.

To Ray, who was an unofficial sponsor for my work.

To my significant other, who continued to support me and reminded me to take a break.

To my colleagues, for sharing in intriguing conversation.

And to my supervisor Paul, for preparing me for a magical world of academia.

Table of Contents

Abstract **ii**

Acknowledgments **iii**

Table of Contents **iv**

List of Tables **vi**

List of Figures **vii**

1 Introduction **1**
1.1 Research Questions 2
1.2 Pillars 3
1.3 Layout 4
1.4 Contributions 4

2 Related Work **6**
2.1 Language Models 6
2.2 Embeddings 9
2.3 Personalization and Domain Adaptation 10
2.4 Personalized Language Models 13
2.5 Targeted Tasks 14
2.5.1 Language Modeling 14
2.5.2 Authorship Verification 16
2.5.3 Word Sense Disambiguation 17

3 Personalized Language Modeling **20**
3.1 Framework 20
3.2 Dataset 21
3.3 Evaluation Measures 23
3.3.1 Adjusted perplexity 24
3.3.2 Accuracy@k 24
3.3.3 Accuracy@k Given c Keystrokes 25
3.4 Language Models 25
3.5 Personalization Techniques 26
3.5.1 Interpolation 26
3.5.2 Continue Training 26
3.5.3 Priming 26
3.6 Results 27

3.6.1 Adjusted Perplexity . . . 27
3.6.2 Accuracy@3 . . . 30
3.6.3 Accuracy@k . . . 32
3.6.4 Accuracy@3 Given c Keystrokes . . . 34
3.6.5 Comparing to RANDOM and DEMOGRAPHIC . . . 35
3.7 Summary . . . 39

4 Authorship Verification **41**
4.1 Framework . . . 41
4.2 Dataset and Evaluation . . . 42
4.2.1 Dataset . . . 42
4.2.2 Evaluation . . . 44
4.3 Methods . . . 45
4.3.1 Embeddings . . . 45
4.3.2 Language Models . . . 45
4.4 Experimental Results . . . 47
4.5 Summary . . . 50

5 Word Sense Disambiguation **51**
5.1 Framework . . . 51
5.2 Data Statement . . . 52
5.2.1 Data selection . . . 52
5.2.2 Annotation . . . 54
5.2.3 Speech Situation . . . 56
5.2.4 Speaker Demographic . . . 56
5.2.5 Dataset Analysis . . . 57
5.3 Methods . . . 58
5.3.1 Baselines . . . 58
5.3.2 SensEmBERT . . . 60
5.3.3 Personalizing SensEmBERT . . . 60
5.4 Experimental Results . . . 61
5.4.1 Tuning SEBERT_PERS_PREDOM . . . 61
5.4.2 Overall Accuracy . . . 62
5.4.3 Author-level Performance . . . 63
5.5 Summary . . . 65

6 Conclusions **67**
6.1 Future Work . . . 68

Bibliography **85**

A Appendix A - Personalized Word Sense Disambiguation Dataset **86**
A.1 REB Form . . . 86
A.2 Author-level Dataset Statistics . . . 97
A.3 Author-level Sense Distributions . . . 97

Vita

List of Tables

3.1 Description of datasets. 23
3.2 Description of language models. 28

5.1 List of lemmas and their frequency for the author who uses the lemma the 10th most frequently (10^{th}MF), and the percent of author-level predominant senses and token senses that are not the grand sense, represented by *Predom* and *Token*, respectively (higher values indicate more diversity). The number of WordNet senses for each lemma under the label *#Senses* is also shown. 54
5.2 The details of the dataset with respect to the number of instances across authors in the dataset and the age of the authors. 56
5.3 The number of instances and assigned senses for each lemma. 57
5.4 Average accuracy across instances (Inst), lemmas (Lem), and authors (Auth). 63

A.1 The details of the dataset with respect to the number of instances per author in the dataset, the age of the authors, and the sex of the authors. 98

List of Figures

3.1 Adjusted perplexity for personalized and non-personalized models that include the neural_background model. (Lower is better.) 28
3.2 Adjusted perplexity for personalized and non-personalized models that include the n-gram_background model. 29
3.3 Accuracy@3 for personalized and non-personalized models that include the neural_background model. (Higher is better.) 30
3.4 Accuracy@3 for personalized and non-personalized models that include the n-gram_background model. 31
3.5 Accuracy@k for personalized models that were tuned on 1000 tokens and non-personalized models that include the neural_background model. (Higher is better.) 32
3.6 Accuracy@k for personalized models that were tuned on 1000 tokens and non-personalized models that include n-gram-based models. 33
3.7 Accuracy@3 given c keystrokes for selected models. 34
3.8 Adjusted perplexity for neural_background and interpolated models. 35
3.9 Accuracy@3 for neural_background and interpolated models. 36
3.10 Adjusted perplexity for neural_background and models that continue training on a second corpus. 36
3.11 Accuracy@3 for neural_background and models that continue training on a second corpus. 37
3.12 Adjusted perplexity for neural_background and neural_background primed on a second corpus. 38
3.13 Accuracy@3 for neural_background and neural_background primed on a second corpus. 39

4.1 Accuracy for each method, using different amounts of training text, and different test document sizes. The number appended to each model's name in the legend indicates the cut-off length for the number of tokens used from the test documents; i.e., *BERT_100* indicates *BERT* with test documents cut-off after the first 100 tokens. (W2V_1000, W2V_10000, and W2V_Full achieve similar accuracies, and therefore, only W2V_1000 can be seen.) . . . 47
4.2 F_1 score for each method, using different amounts of training text, and different test document sizes. 49

5.1 Example of the task assigned to turkers. 55
5.2 Sense distributions for *form*. The average distribution across all authors is in black. 58
5.3 Sense distributions for *charge*. 59
5.4 Sense distributions for four authors for the lemma *deal*. 59

5.5 Average accuracy across authors for the *PREDOM*-based methods compared to *SensEmBERT*. 62
5.6 Author-level performances for the two personalized methods and *SensEmBERT*. 64
5.7 Performance of methods on the x authors that they achieve the lowest accuracy on. 65

A.1 Sense distributions for *case*. 97
A.2 Sense distributions for *charge*. 99
A.3 Sense distributions for *track*. 99
A.4 Sense distributions for *sign*. 100
A.5 Sense distributions for *form*. 100
A.6 Sense distributions for *deal*. 101
A.7 Sense distributions for *rule*. 101
A.8 Sense distributions for *position*. 102
A.9 Sense distributions for *paper*. 102
A.10 Sense distributions for *field*. 103
A.11 Sense distributions for *degree*. 103

Chapter 1

Introduction

Natural language processing (NLP) involves automatically analyzing text written by human authors. Some commonly seen applications from the NLP community include predicting the immediate next word being written in a text, correcting grammar, and translating from a source language to another language. The task itself can vary, but the commonality is that the text is written by a real person in a naturally formed language.

People tend to develop their own use of a language, known as an idiolect, which is unique for each individual. For example, something as simple as the greeting in an email can vary, which can include phrases such as *dear, greetings, hello, hi,* and *good day.* In many NLP tasks, models often do not train on text from a single individual, but they usually train on text that is constructed by combining the text from multiple authors (Welch et al., 2020). This can be used to alleviate the issue of a single author not possessing a sufficient amount of text to train some NLP models, such as training vector representations of words on a Google News corpus (Mikolov et al., 2013) or predicting the sentiment of text by training on a corpus that consists of multiple authors of the same gender (Volkova et al., 2013). I hypothesize that personalized models, which are models that are tailored toward an individual, will outperform models that are not personalized. This would allow models to perform well on individuals and remove the need of grouping authors together. I test my hypothesis by investigating the use of personalization in three different topics, which include language modeling, authorship verification, and word sense disambiguation. Personalization in these fields has received relatively low amounts of attention. To my knowledge, there has not been any work that has studied personalization on the task of

word sense disambiguation.

Language modeling involves estimating the probability for a sequence of words based on the previous words in its context. For example, given the phrase *the cat is up the* _____, we would likely give a high probability to the word *tree* for being the next word. Language modeling has assisted downstream applications such as speech recognition (Young, 1996), machine translation (Och and Ney, 2004), part-of-speech tagging (Ratnaparkhi, 1996), parsing (Choe and Charniak, 2016), and handwriting recognition (Marti and Bunke, 2001). If language models that are tailored toward an individual outperform models that are not tailored toward an individual then they could potentially also assist downstream applications for that individual as well.

Authorship verification involves determining if a given individual is the author of a snippet of text, while only considering text from that individual during training. For example, after observing a set of documents from a single author, a model must predict if any given document belongs to that one author or not. The limitation of allowing models to only observe text from a single author removes the ability to train over negative instances, which eliminates many widely used binary classifiers such as traditional feedforward neural networks.

Word sense disambiguation involves assigning senses — terms that capture dictionary-like entries — to words in context. For example, the term *ball* in the phrase *pass the ball* will most likely be an instance of the sense *round object that is hit or thrown or kicked in games* instead of *a lavish dance requiring formal attire.* Sense inventories contain the set of candidate senses for each lemma — the root of a word. Word sense disambiguation allows the ability to classify the meaning of a word, which can assist downstream applications such as machine translation, plagiarism detection, and next word suggestion on smartphones (Zou et al., 2013).

1.1 Research Questions

This thesis attempts to answer the following three research questions:

1. **Do personalized language models outperform models that are not tailored**

toward an individual? People often write with different styles, and therefore, by considering the individuality of their text, I hypothesize that a personalized language model can perform better for an individual than a generic language model.

2. **Can personalized language models outperform previously state-of-the-art embedding-based models, on the task of authorship verification?** If personalized language models can capture the writing style of an individual then I hypothesize that they can be used to compare the writing style of the individual and the writing style of the text in question to assist in determining if that text belongs to that individual.

3. **Can personalized word sense disambiguation models outperform models that are not tailored toward a single author?** Authors tend to predominantly use one sense per lemma (Gella et al., 2014). This predominant sense, along with sense distributions, can differ between authors. I propose that personalizing a word sense disambiguation system through the inclusion of an author's sense distributions with a state-of-the-art word sense disambiguation model can outperform systems that do not consider author-level information. To study personalized word sense disambiguation, I also propose a novel word sense disambiguation dataset that contains sense annotated instances from a group of authors along with text from those authors.

1.2 Pillars

Along with the research questions, I consider the following pillars when developing and evaluating my models:

1. **Models should require only a relatively low amount of text from any one individual for training.** Many people do not have a large amount of text that can be used to tune a model, and therefore, my proposed models must not require a large amount of text from any one individual. In my experiments, I evaluate models across different amounts of training text — often on a range of 100-100k words. This provides me the ability to study the performance of models when provided with different amounts of text from a single author.

2. **Authors that are included in my experiments should not be selected based**

on their demographics. Models should perform well for authors regardless of their demographics such as age or sex. One factor that all authors have in common is that they have written text in English, which is the common language for the original corpora that I consider when selecting authors for my datasets. In my experiments, I often select authors based on the amount of text that they have available, again, to study the performance of models when provided with different amounts of text from an author.

1.3 Layout

I discuss research that is related to my work in Chapter 2. My experiments are divided into three chapters, where each chapter is dedicated to applying personalization toward a task. In Chapter 3, I focus on personalizing language models. I then recruit those personalized language models and apply them in an authorship verification task in Chapter 4. In Chapter 5, I focus on personalizing word sense disambiguation models. This chapter includes the proposal of a personalized word sense disambiguation dataset as well as personalizing techniques. I conclude my work and discuss potential topics for future work in Chapter 6.

1.4 Contributions

In this work, I focus on personalizing models for three different topics, including language modeling, authorship verification, and word sense disambiguation. There has been a relatively small amount of work done involving personalization in these fields. In the topic of language modeling, I propose methods for personalizing language models, while requiring a relatively small amount of text from an individual. Specifically, I tailor a long short-term memory language model by allowing it to see a small amount of text from an author and then freezing its state before estimating the probabilities of words over a sequence. My findings indicate that this personalized method improves the performance of language modeling systems. On the topic of authorship verification, I propose a model that compares a personalized language model against a non-personalized language model to determine if a snippet of text is written by a selected author. I show that my personalized language models-based approach outperforms the previously state-of-the-art embedding-based meth-

ods. On the topic of word sense disambiguation, I propose a novel dataset that includes sense annotated instances of nouns from a group of authors and also provides unannotated text from those authors. I also propose a personalized word sense disambiguation approach that extends a state-of-the-art model (Scarlini et al., 2020) by introducing sense distributions from the author. I show that personalized word sense disambiguation models can outperform models that are not personalized. To my knowledge, my work is the first to study personalization on the task of word sense disambiguation.

Chapter 2

Related Work

In this chapter, I describe existing work that focuses on generic language modeling and embeddings, before discussing methods that integrate information about individuals or their demographic. I then discuss work in the more specific tasks (language modeling, authorship verification, and word sense disambiguation) that become the main focus of this dissertation.

2.1 Language Models

Language modeling involves models that estimate the probability for a word in a snippet of text based on the words in its context — often limiting the context to words that occur before it. In doing so, the model estimates probabilities for entire sequences of words. These models are known as language models. The snippets of text that they utilize can range from a few words before a word up to spanning across sentences and documents. Language models have assisted downstream applications including speech recognition (Young, 1996), machine translation (Och and Ney, 2004), part-of-speech tagging (Ratnaparkhi, 1996), parsing (Choe and Charniak, 2016), and handwriting recognition (Marti and Bunke, 2001). Language models have also been used in text classification tasks, which involves classifying a piece of text as one of a set of predefined classes. This includes sentiment analysis, question classification, and topic classification (Howard and Ruder, 2018), and determining if a tweet is relevant to a natural disaster (Alam et al., 2018). Common types

of language models include traditional n-gram models, which were used in Benedi and Sánchez (2000), Zhang and Dong (2003), and Heafield et al. (2013). N-gram models are statistical models that calculate the probability of a word in a sentence given the previous $n-1$ words. Recently, neural language models have been the focus, when considering a language model, due to their superior performance over n-gram language models (Mikolov et al., 2010). Neural language models use a neural network to estimate the probabilities for words in context. The inputs to the neural language model are the words that precede the target word that the model is predicting and the output is a probability distribution for all words in the model's vocabulary. Recurrent neural networks (RNN) are often recruited for neural language models, which allow the ability to preserve word ordering. Largely, long short-term memory (LSTM) language models were favored due to them not requiring a fixed amount of input and their ability to weight how much consideration is given to previously seen words in a document (Sundermeyer et al., 2012). Recently, transformer-based language models (Radford et al., 2019) have been achieving state-of-the-art performance on tasks such as question answering and natural language understanding (Devlin et al., 2019). At the core, transformers consist of a series of neural-based encoders and decoders with attention mechanisms between layers. Encoders are models that take an input sequence — a sequence of words in this case — and map it to a dense vector representation. Decoders are trained to perform a mapping from the previously mentioned vector to the original input sequence.

Groups have augmented neural language models to achieve better performance. These techniques could also be applied to personalizing language models or be used as strong baselines. Salle and Villavicencio (2018) reserve certain hidden weights of their LSTM language model for frequent words and another set of weights for infrequent words. This allows the model to preserve more information about frequent words, which would have been seen in more contexts. They trained and evaluated their model on the Penn Treebank dataset (Marcus et al., 1994) — an annotated collection of text from the Wall Street Journal.[1] The frequent words used in the model could pertain to the frequent words used by a single person. Therefore, the model could focus more on words that are frequently

[1] `https://www.wsj.com`

used by the person. Similarly, Mikolov and Zweig (2012) modify a recurrent neural network (RNN) by adding a feature vector that is connected to the hidden layer and output layer. They include the topic in this feature vector to avoid the issue of remembering long distance information pertaining to the topic. The topic vector is a topic distribution estimated by a latent Dirichlet allocation model (Blei et al., 2003) on a fixed number of words before the current word. Because the process of estimating the distribution of topics is expensive, they represent it by multiplying the topic distribution for each word in the set of words before the target word. They evaluated using perplexity on the Penn Treebank dataset (Marcus et al., 1994) and word error rate on the Wall Street Journal dataset (Paul and Baker, 1992). The feature vector could represent a single person instead of a topic — making the model personalized to that person. Devlin et al. (2019) proposed the Bidirectional Encoder Representations from Transformers (BERT) model, which introduces the use of a masked language model objective. This means that during training, tokens in the input are randomly redacted (masked) and the language model is scored based on its ability to predict the word that was masked. BERT is trained using words that occur from before and after the masked token. This alleviates the transformer's dependence on direction while still considering the positions of tokens.

Training models for language modeling involves tuning parameters such as the number of words to consider from the context of the target word. For example, if the model is needed to predict a word at the end of a 1000 page book, then the model does not need to see the entire first section of the book as input. Whenever training language models, regardless of their desired tasks, it is essential to consider which parameters will result in language models that achieve a strong performance. For tuning parameters of LSTM language models in my experiments, I turn to Khandelwal et al. (2018), who observed the effects of varying different parameters of a regular LSTM model. They did an ablation study on the Penn Treebank dataset (Marcus et al., 1994) and WikiText-2.[2] The ablation is done by giving the LSTM different lengths of text and replacing, removing, or switching words. They found that LSTM models could distinguish between the previous 50 words when inputted with text longer than a sentence and are capable of using almost 200

[2] `https://blog.einstein.ai/the-wikitext-long-term-dependency-language-modeling-dataset`

previous words of context, but the model only focuses on the closest 50. Words further away are more likely to be grouped and treated as a topic. They found that word order is significant in nearby words but less so in far away words. This gives an estimate on how many preceding words to consider for LSTM language models. Due to the cost of training transformers, it is common to use pretrained models and tune them for a specific task. Therefore, the training parameters are set by default in the pretrained model.

2.2 Embeddings

Embeddings are dense vector representations of text, such as a word, sentence, or document. Models that produce embeddings are often trained using language models or neural networks similar to language models. The benefit of embedding representations over their predecessor (one-hot encoding) is that embeddings exist in a continuous vector space, where similarity between embeddings can be utilized. Bengio et al. (2003) proposed the concept of training embeddings using neural networks. Mikolov et al. (2013) extended this idea by decreasing the complexity of the model through estimation and resulted with the popular models known as word2vec's continuous bag of words (CBOW) and skip-gram. These models train word embeddings using a feedforward neural network. Similar to language models, CBOW estimates a probability for a target word given words from its context. Instead of restricting the model to only the words that precede the target word, the input to CBOW is the words within a window that spans before and after the target word. Skip-gram trains its word embeddings by predicting the context words within the window, given a target word. Word embeddings have performed well on tasks such as the Microsoft sentence completion challenge (Melamud et al., 2016),[3] word similarity (Huang et al., 2012; Luong et al., 2013), and word analogy (Mikolov et al., 2013). Larger amounts of text have been embedded, such as a fixed context around a word (Melamud et al., 2016), a sentence (Kiros et al., 2015), and a document (Le and Mikolov, 2014). Unlike word embedding models, these models generate embeddings given an instance of text, whereas word embeddings are static vectors after their training is complete. Transformer-based models achieve

[3]The task was originally proposed by Zweig and Burges (2012).

strong results in tasks while not being bound to one type of embedding (Devlin et al., 2019). For example, they contain static word embeddings while also having the ability to embed a given sentence or document.

User-level word embeddings are word embeddings that are trained for individuals. Lynn et al. (2017) trained them by performing a factorization on a user-word occurrence matrix and were able achieve better performance for individual users on tasks such as part-of-speech tagging, prepositional phrase attachment, sentiment analysis, sarcasm detection, and stance detection. Ebrahimi and Dou (2016) trained user-level word embeddings by maximizing the probability of a word given the user and generic word embeddings and have achieved better performance on sentiment classification and authorship attribution. User embeddings are similar to user-level word embeddings except that they represent a single user instead of words from a single user. Ebrahimi and Dou (2016) train user embeddings alongside word embeddings using a log-bilinear model and get a user-specific probability distribution over words by multiplying them together. Amir et al. (2016) generated user embeddings by maximizing the probability of a sentence given the user. The increase of performance on tasks with embeddings that consider individuals suggests that other NLP systems should also be tuned toward individuals, which is discussed in the following section.

2.3 Personalization and Domain Adaptation

Personalization of an NLP model involves bettering its performance for an individual. For example, a personalized sentiment analysis model might predict different sentiments for the snippet, *That movie was sick*, with the word *sick* being associated with a positive or negative sentiment, depending on the individual who wrote it. Flek (2020) proposes a shift of focus towards personalized models over generic models, to better represent, study, and evaluate text from individual users. Often, models do not have access to a large amount of text from a single individual, and therefore, it can be difficult to train a personalized model using traditional methods. A common approach to overcome this low-resource problem is to tune the model to perform better for a group of people based on some demographic. Volkova et al. (2013) used the gender of the authors to assist in sentiment analysis. Hovy (2015)

found that including information of the gender and age of the author can assist models performing classifications involving sentiment and topics. They also used one attribute of the author, i.e., age or gender, to estimate another attribute of the author, such as gender or age, respectively. Unfortunately, models may not have access to metadata about the person due to privacy concerns or limitations of the social media platform on which the text exists. To overcome this constraint, Huang and Paul (2019) induced the personal traits of the author to improve document classification. Although this is a way to utilize unstated traits of the author, the overall model becomes reliant on the performance of the trait classification and requires the predicted trait to be present in training data to allow the trait classifier to be able to assign the trait. Therefore, if the trait classifier performs poorly, then the overall model will perform poorly. Following a similar logic of adapting to a demographic, domain adaptation can also be used to assist demographics/domains that do not contain a relatively large amount of a text.

A domain is a source of text that exhibits some commonality such as a social media platform or newspaper section. For example, the *sports* and *business* sections of a newspaper would be two different domains. Domain adaptation involves adapting a model that was trained on a domain to a different domain, which often contains a relatively smaller amount of text and data. Personalizing language models can be viewed as a domain adaptation problem, where text from a specific user is the domain that the models must adapt to. Domain adaptation techniques have been applied to tasks in other areas such as adapting language models to YouTube video categories (Irie et al., 2018), adapting language models to different languages using cross-lingual embeddings (Adams et al., 2017), and adapting query completion models to different newspaper sections (Jaech and Ostendorf, 2018). Alam et al. (2018) used domain adaptation to classify tweets as being relevant or not relevant to a disaster — in this case, the disaster was an earthquake. The model is adapted from working with a previous disaster to a current disaster using a neural network to embed the tweet, which is used in a classifier to predict if the tweet is relevant and another classifier to determine the event involved in the tweet, such as a specific disaster. Similarly, Irie et al. (2018) approached domain adaptation between YouTube videos' categories with multiple LSTMs. Each LSTM is domain-specific, which represents a specific category,

and are combined with a "mixer" LSTM that determines how much weight to give to each domain-specific LSTM. Howard and Ruder (2018) train a language model on a large general domain corpus and then tune it further for a specific task. They initially train their LSTM language model on Wikipedia articles before fine-tuning their model to the target domain using one of two methods: discriminative fine-tuning (different layers get different parameters) and slanted triangular fine-tuning (learning rate is increased by a large amount near the start of training and then reduced over time). After the language model is trained and tuned, they then train the target task classifier, which uses an activation layer and a softmax to estimate a probability distribution across classes. The classifier is the only part that is trained from scratch for each classification task. Their method outperforms state-of-the-art models on sentiment analysis, question classification, and topic classification.

Word embeddings have been used to adapt language models to a specific domain. Zhang et al. (2016) interpolate their general word embeddings with pretrained domain-specific word embeddings to better the performance of a language model within a domain. Irie et al. (2018) use word embeddings that were trained on a non-domain-specific background corpus to initialize the word embeddings for domain-specific LSTMs before they are further trained on a domain-specific corpus. This allows the domain-specific LSTMs to have better representations of words before they begin to train on domain-specific text. The domain-specific text often does not contain a relatively large amount of data that is needed to train an LSTM to the point that it achieves a high performance. Adams et al. (2017) apply this idea to cross-lingual language modeling by treating text from one language as a background corpus and another language as the domain that is being adapted to. Welch et al. (2020) train demographic-based word embeddings by jointly training both generic word embeddings and word embeddings that are specific to each demographic, which includes age, gender, religion, and location.

The setting in which text is written is known as the register. For example, the language one might use in a document written to their colleague would be quite different than the language used in a document written to a person in a more intimate relationship with the author. Laippala et al. (2020) and Egbert et al. (2015) focus on work that annotates text with register classes for online content. The possible categories of registers that they

include are, but not limited to, fiction, discussion, and technical information. The register of text can also consider the addressee and can be used to model the language of a person while considering who they are conversing with (Li et al., 2016).

Teevan et al. (2005) approached the problem of information retrieval, while considering personalization for the individual that is submitting a query. They found that by considering the content that the user has seen/written, they were able to achieve a better performing information retrieval system. The type of content that they considered includes the text that the individual has written or read in emails, previously visited Web pages, and previously submitted search queries.

2.4 Personalized Language Models

It is reasonable to assume that personalizing language models will assist applications that use language models such as speech recognition (Chelba and Jelinek, 2000), machine translating (Och and Ney, 2004), part-of-speech tagging (Ratnaparkhi, 1996), parsing (Choe and Charniak, 2016), and handwriting recognition (Plötz and Fink, 2009) to perform better for individual users. In order to personalize language models, some groups have used user embeddings. Jaech and Ostendorf (2017) propose FactorCell, which modifies the recurrent layer of their RNN language model with a personalized matrix that is generated based on metadata about the user (Amir et al., 2016; Jaech and Ostendorf, 2018; Lynn et al., 2017). Jaech and Ostendorf (2018) also modified the recurrent layer for personalized language models. They approach the task of query completion, which involves finishing a word or phrase given a prefix, with character-level language models. They train user embeddings in an end-to-end setup and unseen user embeddings are learned online. In this case, the recurrent layer is augmented with the trained user embeddings instead of raw metadata about the user. Similarly, Mikolov and Zweig (2012) also used a character-level language model, but they concatenated a user embedding with the current character embedding at each timestep to achieve personalization. Ji et al. (2019) and McMahan et al. (2017) use a framework that involves sharing language model parameters between users' local machines and a public server to preserve privacy for each user. In this framework, a language model

on each user's machine and a general language model are jointly trained with only parameters of the language models being shared and not text. Yoon et al. (2017) compare three different methods for personalizing LSTM language models: training an LSTM solely on text from a user; training an LSTM on text from a collection of users and then freezing it and adding a layer to the LSTM for training on user-specific text; and training an LSTM on text from a collection of users and then freezing a few layers while tuning others on user-specific text. They found that the two methods that involve freezing components of the LSTM achieve superior performance.

Michel and Neubig (2018) apply user adaptation techniques to machine translation by treating each language as its own domain. They add a user-specific word at the beginning of each sentence from that user to prime a language model for that user — personalizing the language model for them for that sentence. They use a bias based on the user's vocabulary for the softmax equation and collect all users' biases and factor them to give a low dimensional user vector. A model can leverage metadata about the author such as age, gender, and personality, to assist in personalized-NLP tasks (Lynn et al., 2017). Li et al. (2016) used metadata about the author along with two LSTM models with one model being used to model an author and another model that is used to represent how the author's text changes given a specific addressee. Although metadata can be helpful, it is often not available.

2.5 Targeted Tasks

In this section, I discuss the three tasks that will be the main focus of my work. They are presented here in the same order that they will be discussed in the remaining parts of this dissertation.

2.5.1 Language Modeling

Language model evaluation involves measuring a language model's ability to predict the next word in a sentence, given its context words. The most common metric used for this evaluation is perplexity, which is defined as $\mathrm{PPL} = -\frac{1}{N}\sum_{i=1}^{N}\log(\mathrm{p}(\mathrm{w}_i))$ with w_i

being the correct word at position i, and N being the number of words in the text that are being used for evaluation. This shows that higher probabilities of the correct words will produce lower perplexity, and therefore, a better score. A limitation of this metric is that a fair comparison on a fixed corpus requires all language models to contain the same vocabulary due to the use of the unknown token, often written as *unk*. Due to language models requiring the ability to handle all words that they may see, *unk* is used to replace all words that were not observed during the language model's training phase. The evaluation method does not change when encountering *unk* and it is treated the same as other words in the model's vocabulary. The word *unk* can also be used during training to replace low-frequency words to decrease the complexity of the training and learn better estimates for *unk*. Both usages of *unk* alter the size of a language model's vocabulary and therefore affect the probability distribution outputted from the model. For example, if language model LM_a had 100 words in its vocabulary and language model LM_b had 1000 words in its vocabulary, then LM_a has far fewer words that it needs to consider when distributing weight, making it an easier evaluation for it. For this reason, Ueberla (1994) proposed adjusted perplexity, which penalizes models when they predict *unk*, recalculating the probability of *unk* as $\text{p(UNK)} = \frac{\text{p(UNK)}}{|\text{UNK-TYPES}|}$, where $|UNK\text{-}TYPES|$ is the number of word types that have been replaced with *unk*, and therefore, penalizes a system that replaces many word types with *unk*. A word type represents all occurrences of that word. Another evaluation metric that can be used is word error rate, which involves measuring the number of words that a model incorrectly labels as being the most probable word. The final evaluation that I discuss is accuracy at k, which involves calculating if the correct word is among the top k words in a list ranked by probability. Accuracy at k is an intrinsic evaluation that is influenced by the desired performance of the extrinsic evaluation known as next-word-prediction, which is commonly seen on smartphones' messaging software, with k often being 3. Intrinsic evaluations involve methods to evaluate models in their own environment, while extrinsic evaluations measure the performance of a model within an application that is often interacted with by some user. Intrinsic evaluations are useful for directly comparing methods that are being applied to the same problem, while extrinsic evaluations show how useful a method can be in an application.

Due to the nature of the task, with the correct labels being the words already present in the text, language modeling datasets can easily be obtained from a wide range of domains and languages, while requiring little to no manual annotations. Commonly used corpora are often divided into three predefined sets including a training set, validation set, and test set. Having the sets predefined allows for different models to compare without rerunning previously established models. A commonly used benchmark dataset is the Penn Treebank dataset (Marcus et al., 1993), which has been divided into a training set (929k words), validation set (73k words), and test set (82k words). Other frequently used datasets include WikiText-2 and Wikitext-103 (Merity et al., 2016), and the Google Billion Word benchmark (Chelba et al., 2013). Schler et al. (2006) produced a large corpus from blog text that contains roughly 300 million words from approximately 71k people. This dataset was originally constructed for studying the effects of age and gender on text, which is provided in the dataset.

2.5.2 Authorship Verification

Authorship attribution involves determining the author of a snippet of text from a list of candidate authors. Similarly, authorship verification focuses on if a given author has written a selected snippet. This is useful in the field of forensics to identify a guilty party (Luyckx and Daelemans, 2008) or to determine the legitimacy of a snippet of text. For example, media posts from a public icon's account defacing themselves might be an indicator that they were not the true author. Authorship verification has been applied to domains such as emails (Schmid et al., 2015), texts from university students (van Halteren, 2004), and dialog from plays and online opinion articles (Stamatatos et al., 2015). Unlike authorship attribution, which involves the ability to observe text from other authors, authorship verification focuses only on text from the single author, limiting its ability to train using negative samples.

The organization PAN released an authorship verification shared task in 2013 (Stamatatos et al., 2013). Two of the models submitted include one using n-grams and another using KNN classifiers with word prefixes and suffixes, parts-of-speech, and character n-grams as features. Jankowska et al. (2014) performed well on the 2013 task and was used as the

benchmark for the PAN authorship verification task of 2014.[4] They classified a document as belonging to the target author based on word frequency similarity between it and documents that are known to belong to the author. In early work, Luyckx and Daelemans (2008) use a Chi-square metric to select features that would best differentiate documents from a set of features including part-of-speech, n-grams, and function words. The selected features are then used as input for an SVM.

2.5.3 Word Sense Disambiguation

The meaning of a token — a word in context — depends on the context that it occurs in. Each token can be classified under word senses, which capture dictionary-like definitions. These senses are defined by humans, and therefore, vary between dictionaries known as sense inventories. Three inventories that have been used include WordNet (Miller, 1995), which additionally represents words in a graph-based manner according to similarity; OntoNotes (Hovy et al., 2006), which includes the structure of an ontology; and the MacMillan dictionary,[5] which contains more coarse-grained senses. Assigning word senses to tokens is called Word sense disambiguation (WSD).

Two different frameworks for WSD tasks include knowledge-based, which utilizes information contained in a sense inventory; and supervised, which involves training on annotated instances. Many modern knowledge-based methods extend the simplified Lesk method (Kilgarriff and Rosenzweig, 2000), which involves classifying a token with the sense that contains the most overlapping words in its definition with the context of the word being classified. WSD methods that involved a Lesk-style approach are Banerjee and Pedersen (2002), which includes looking at the definition of words that are similar to the token being classified according to WordNet and Basile et al. (2014), which looks at the definition of similar words but adds a vector representation of the target token — generated by a topic model — into their similarity calculations. Topic modeling is a probabilistic model that views documents as a distribution over topics and topics as a distribution over types (Blei et al., 2003). Topic modeling was also applied to WSD in Boyd-Graber et al. (2007) and

[4] `http://pan.webis.de/clef14/pan14-web/author-identification.html`
[5] `https://www.macmillandictionary.com`

Li et al. (2010).

The current state-of-the-art knowledge-based model (Scarlini et al., 2020) creates an embedding for each sense of a word by using BERT to embed both the sentences in Wikipedia articles related to that sense — determined by words in the sense's synset — and the gloss of the sense. The two vectors are then concatenated to make the final sense embedding. At runtime, they embed the context of the target word using BERT and compare similarities to the previously described sense embeddings to determine which sense is assigned. This model also has the added bonus of performing well on rare words and senses (Scarlini et al., 2020). This is important to consider for my experiments with author-specific text since authors tend to favour a single sense for a word (Gella et al., 2014), which could differ among authors. Therefore, if a model performs poorly on rare senses according to the sense inventory, then it might perform poorly on the favoured sense of the author. Likewise, if an author frequently uses rare words — according to a corpus — then a model that performs poorly on rare words would perform poorly for the author.

Raganato et al. (2017) applied supervised models, which involved embedding the target word and using it as input to an SVM along with other features such as the part-of-speech of words in the context of the target word. The current state-of-the-art supervised model embeds the context of the target word using BERT, which is used as input, along with edge weights from Wordnet, to a feedforward neural network to perform the sense classification (Bevilacqua and Navigli, 2020).

Two baselines used in WSD tasks are the first sense heuristic and the predominant sense baseline (Gale et al., 1992). The first sense heuristic involves labeling each word with its first sense from a sense inventory, which is usually the most frequently observed sense in the data used to construct the sense inventory. The predominant sense baseline, which is supported by the observation of the one-sense-per-discourse heuristic — assigns the sense that has the highest frequency within a specific domain (McCarthy et al., 2004). The one-sense-per-discourse heuristic suggests that there is a sense for each word that is far more frequent for a specific document. Gella et al. (2014) found that the one-sense-per-discourse heuristic applies to Twitter users — where all tweets from a given user are viewed as a single discourse — with each user often favoring one sense for a particular word. The

usefulness of knowing predominant senses has led to the task of predicting predominant senses (McCarthy et al., 2004; Koeling et al., 2005; McCarthy et al., 2007; Lau et al., 2014). McCarthy et al. (2004), Koeling et al. (2005), and McCarthy et al. (2007) use a similarity metric to map a document containing numerous instances of a word type to a WordNet sense distribution or ranking of senses in order to predict the predominant senses of words. McCarthy et al. (2007) also found that nouns exhibit their predominant sense more strongly than verbs. Lau et al. (2014) estimate a sense distribution for a single word using a topic model, which is then aligned with a sense inventory and the sense that is given the most weight is classified as the predominant sense.

I propose a novel dataset for WSD and therefore, I will discuss existing datasets. One of the more popular English WSD datasets is a collection of existing datasets and sense-annotated corpora (Raganato et al., 2017), which was created to facilitate the comparison of models' performances. The evaluation datasets within this unified framework include datasets from five shared tasks, which are Senseval-2 (Edmonds and Cotton, 2001), Senseval-3 task 1 (Snyder and Palmer, 2004), SemEval-07 task 17 (Pradhan et al., 2007), SemEval-13 task 12 (Navigli et al., 2013), and SemEval-15 task 13 (Moro and Navigli, 2015). The two sense-annotated training corpora include SemCor (Miller et al., 1994) and One Million Sense-Tagged instances (OMSTI) (Taghipour and Ng, 2015). The dataset that I propose considers the author of the text being annotated, and therefore, I turn to the work of Gella et al. (2014), which focuses on real tweets from individuals. They create a sense annotated dataset that consists of 20 authors and 20 nouns. Interestingly, they show that the authors exhibit a one sense per discourse trait, meaning that an author tends to favour a single sense for a given noun. This finding suggests that considering the predominant sense of an individual or their sense distribution might assist WSD models. I use the work from Gella et al. (2014) as a guideline, when considering the creation of my dataset.

Chapter 3

Personalized Language Modeling

In this chapter, I discuss my approaches to personalizing language models for a single person, while requiring relatively low amounts of text from them. This work is an extended version of King and Cook (2019) and King and Cook (2020b).

3.1 Framework

In this work, I consider the problem of personalizing language models, that is, building language models that are tailored to the writing style of an individual. Language modeling often requires a large amount of text to sufficiently train a traditional language model. However, a large amount of text from any one user — for example social media users such as bloggers — is not necessarily available for training a user-specific language model. To overcome this issue, some techniques treat text from other users that share demographic characteristics, such as age or gender, as training data. Although this has been shown to improve the performance of language models (Lynn et al., 2017), this demographic information is typically obtained via document metadata, which is not available for all document types, or for all users. This suggests a need to personalize language models while only relying on a small amount of text from the user.

I propose and evaluate three different personalization techniques, while limiting the models' access to different amounts of text from the user. These techniques are applied to a language model that was trained on a large background corpus of in-domain text (specifi-

cally blog posts). The three techniques include continuing to train the background model on text from the user, interpolating the background model with a model that was trained on the user's text, and priming the state of the background model with text from the user. My results show that I am able to improve the performance of a background long short-term memory (LSTM) neural network language model using any of the mentioned personalization techniques. The interpolated models perform best when a relatively large amount of text is available from the user, however, priming models performs best when only a small amount of user text is available. I further apply the interpolation technique to an n-gram background language model and show that it improves its performance, similarly to how it improved the LSTM language model.

Furthermore, I benchmark my personalized models against models that use the same techniques, but that use text from users with the same demographic characteristics as the target user — specifically age and gender — or text from randomly-selected users. This can be seen as a model adaptation technique, where the model is being adapted to a group of users. My findings indicate that all three of my proposed approaches to personalization out-perform model adaptation based on demographics and randomly selected users, and that these improvements are not the result of the language model simply having access to more data.

3.2 Dataset

The dataset that was used consists of blog posts from 19,320 users. It was originally used in Schler et al. (2006). I perform sentence splitting on each document, and add start-of-sentence and end-of-sentence tokens to each sentence. These are special tokens (often indicated as <sos>and <eos>) that indicate where a sentence begins and ends, respectively. For example, *<sos>the cat climbed a tree. <eos>* might be a sentence in the dataset.

I select the 10 users who have the largest amount of text to form my set of users that I use for evaluation, which I label *USER*. My test users consist of 6 males and 4 females with ages ranging from 14 to 47 years old with a mean of approximately 28 years. I split

up the text from each user in *USER* into a training set and a held out set using a 75/25 split, without splitting up blog posts. I do not split up blogs to ensure that I do not have parts of the same blog post in the training and testing data. I randomly select sentences from the training set so that each user has 6 corpora with the following numbers of tokens: 100, 500, 1000, 10k, 100k, and all of the text available for that user, which ranges between $226k$ and $446k$ tokens. I refer to these corpora as *USER-TRAIN*. The held out set consists of approximately 25% of the user's text, from which I randomly extract sentences until I have at least 10k tokens from each user. I refer to this test set as *USER-TEST*. I use *USER-TRAIN* to build personalized language models, and test them on *USER-TEST*.

I use the 20 users with the largest amount of text after the top 10 to form my random samples, which are used for creating a dataset that is not limited to text from a single user. This is used to compare a model that applied personalizing techniques on text from multiple users against a model that applies the same personalizing techniques on text from a single user. To create this dataset, I randomly extract samples of text of the same sizes as *USER-TRAIN* from this group of 10 users to form this dataset, referred to as *RANDOM*. The text from the remaining 19,290 users is used to build my background corpus (*BACKGROUND*). The text from any one user is limited to 30k tokens to avoid text from one user saturating the corpus and biasing the corpus to strongly represent their text. I replace each type in *BACKGROUND* that has a frequency less than 10 with a special *UNK* token. I do this to reduce the cost of training language models. The resulting corpus contains approximately 116M tokens and 93k word types. A token is any instance of a word in context and a word type represents all instances of a word.

I randomly select 5 users from *RANDOM* for tuning my models. I split up the text into training (*TUNE-TRAIN*) and testing (*TUNE-TEST*) sets, where *TUNE-TRAIN* contains 1000 tokens from each user and *TUNE-TEST* contains the remaining tokens.

I randomly select sentences from non-test users (i.e., users not in the *USER* set) that have a common age and gender with the target user, to generate 5 demographic-based corpora per user in *USER* with different amounts of text to form the set *DEMOGRAPHIC*. The sizes of the corpora for each user are similar to those for *USER-TRAIN*, i.e., roughly 100, 500, 1000, 10k, and 100k tokens. The number of users that have the same age and gender

Name	Definition
BACKGROUND	Blogs from 19,290 users used to train background models
USER-TRAIN	Samples of text of varying sizes from each of 10 different users; used for training models
USER-TEST	10k tokens from each of the 10 users from USER-TRAIN; used for testing models
TUNE-TRAIN	Tokens from each of 5 different users; similar to USER-TRAIN but for tuning models
TUNE-TEST	Varying amounts of text from each of the 5 users in TUNE-TRAIN; used for evaluation during model tuning
RANDOM	Text from 20 users with the next largest amount of text after those in USER-TRAIN and USER-TEST; used as random text for benchmarking
DEMOGRAPHIC	Text from users that share the same age and gender as each user from USER-TRAIN; contains samples of the same amounts of text as USER-TRAIN

Table 3.1: Description of datasets.

as the target user ranges from 45 (females aged 47 years) to 2380 (males aged 17 years). I perform the same preprocessing on all corpora, which involves casefolding, tokenization by the Stanford Core NLP toolkit Manning et al. (2014), and converting all numerals to a special <*num*> token. A summary of the datasets is shown in Table 3.1.

3.3 Evaluation Measures

I evaluate my models on each of the 10 users from *USER-TEST*. I use the text from *USER-TRAIN* that corresponds to the same user to personalize a language model trained on *BACKGROUND*. For example, I use text from user x in *USER-TRAIN* to personalize my model, which is evaluated on text from user x in *USER-TEST*. I allow my models to use different amounts of text from *USER-TRAIN*, which includes 100, 500, 1000, 10k, and 100k tokens, and the full amount of the text available. I test my models at the sentence level, which means that the language models cannot use information from previous sentences, and that the cell state of the LSTM is reset before each test sentence so that the results are not affected by the ordering of the test sentences. I explain the three evaluation metrics considered in the following subsections.

3.3.1 Adjusted perplexity

Perplexity is defined in Equation 3.1, where N is the number of tokens in the corpus and $p(w_i)$ is the probability of word w at position i.

$$\text{perp} = -\frac{1}{N}\sum_{i=1}^{N}\log(\text{p}(\text{w}_i)) \tag{3.1}$$

It is one of the most common ways to evaluate language models when each model contains the same vocabulary, which is the list of words that a model has seen. Since perplexity can't be used to compare language models with different vocabularies and the personalization techniques considered can alter the vocabulary, I use adjusted perplexity (Ueberla, 1994) as an evaluation metric. Language models are evaluated on their ability to estimate word probabilities and if a model has fewer words that it needs to consider when performing the probability estimations, then it is given an unfair advantage. Therefore, adjusted perplexity penalizes language models when the target word type is not in the vocabulary, which means that the token is replaced with the special *UNK* token. The equation for adjusted perplexity is the same as perplexity, with the exception that the probability of *UNK* is recalculated using Equation 3.2 below:

$$\text{p}(\mathit{UNK}) = \frac{\text{p}(\mathit{UNK})}{|\mathit{UNK\text{-}TYPES}|} \tag{3.2}$$

with *UNK-TYPES* being the set of types that are converted to *UNK* in the test data.

3.3.2 Accuracy@k

To evaluate my models on a more extrinsic setup, I evaluate using accuracy@k for each token in the test set. This metric says a language model predicted correctly if the next word in the corpus is in the top k words predicted by the model. It reflects the desired behaviour of a next-word suggestion feature, as on many smartphone soft keyboards. Word-error rate is another common metric used for evaluating language models but does not capture the ranking of predicted words which is often seen in a next-word suggestion system. I exclude the end-of-sentence marker from this evaluation because it would not be part of a text

being typed. The start-of-sentence token is used as the first input to the language model, but the first predicted position is the token after the start-of-sentence token.

3.3.3 Accuracy@k Given c Keystrokes

This metric extends accuracy@k by allowing the model to see the first c characters of the target token. If c is greater than or equal to the length of the target token, then the probability for the target token is set to 1. Again, this evaluation metric lends itself to the desired behavior of a language model that is assisting a person who is writing a message, and has typed the first c keystrokes of the next word. Similarly to accuracy@k, I exclude the end-of-sentence marker from this evaluation.

3.4 Language Models

I explore two different types of language models in my experiments. The first model is an n-gram language model that uses Kneser-Ney smoothing (Heafield et al., 2013) with $n = 3$. Preliminary experiments showed that $n = 3$ yielded the lowest adjusted perplexity on *TUNE-TEST*.

The second model is a neural language model that uses an LSTM. I performed preliminary experiments by testing on *TUNE-TEST* to tune my LSTM models. I performed a grid search over the following parameters with their associated values: number of hidden units $(128, 256, 512, 1024, 2048)$; number of layers $(1, 2)$; size of embeddings $(64, 128, 256, 512, 1024)$; number of training epochs $(1, 2, 3)$; and batch size $(1, 2, 5, 15, 30, 45)$. I found that the best performing neural language models that were trained on *TUNE-TRAIN* and tested on *TUNE-TEST* contained 256 hidden units, an embedding size of 256, contained 1 layer, and trained for 1 epoch with a batch size of 2. I used the default settings for the neural language model that was trained on the background corpus which was an embeddings size of 128 with 1024 hidden units, and 1 layer, which was trained using a batch size of 45 for 1 training epoch. The batch size for the user-level model is smaller than the model trained on the background corpus, because the training data for each user is relatively small. For all LSTMs, I set the sequence length to 30, the

learning rate to 0.002, and use cross entropy as the loss function.

3.5 Personalization Techniques

I discuss the three different personalization techniques for this experiment in the following subsections.

3.5.1 Interpolation

In this approach, I interpolate a language model that was trained on *BACKGROUND* with a language model that was trained on *USER-TRAIN* by calculating the element-wise mean of their probability distributions. If only one of the two language models' vocabularies contains a given type, then the language model that is missing the type gives a probability of 0. Therefore, the interpolated probability for that type would be $(p+0)/2$, where p is the probability from the language model that contains that type. Since the n-gram models perform relatively poor compared to the neural models, this is the only personalization technique that is applied to the n-gram model that was trained on *BACKGROUND*, discussed in Section 3.6.1.

3.5.2 Continue Training

This personalization technique allows a neural language model that was trained on *BACKGROUND* to continue training on *USER-TRAIN*. I do not update the vocabulary of the language model, which means that all out-of-vocabulary words — words in *USER-TRAIN* that are out-of-vocabulary with respect to *BACKGROUND* — will be replaced with *UNK*. I set the batch size to 2 for the training on *USER-TRAIN* (similar to the models trained only on user-level text) because of the small amount of text that is used for training. All other parameters remain the same.

3.5.3 Priming

In this setup, I prime a language model that was trained on *BACKGROUND* by inputting tokens from *USER-TRAIN*, which updates the cell state of the language model without

changing the weights of the LSTM. I then freeze the state of the language model and perform testing as usual. Preliminary experiments showed that primed language models achieved poorer performance on longer snippets of text, which I believe is due to the language model losing its initially primed state. To mediate this, I reinitialize the state to the frozen primed state before each sentence in the test data.

3.6 Results

In this section, I show results for my experiments in terms of adjusted perplexity (Section 3.6.1), accuracy@3 (Section 3.6.2), accuracy@k (Section 3.6.3), and accuracy@k given c (Section 3.6.4). I then compare personalized models against models that use the same techniques, but with text from users that share demographic characteristics with the target user, and with randomly-selected users (Section 3.6.5).

I separate the results based on the background model and the evaluation metric. All graphs show results over different amounts of text from the secondary corpus, except when using accuracy@k and accuracy@k given c, where I show results for different values of c. The background models are all trained on the entire amount of text from *BACKGROUND*. The models are named using their model type and personalization technique. For example, the neural model that was trained on *BACKGROUND* and then primed on text from the user would be called neural_background_primed_user. A description of the models is shown in Table 3.2.

3.6.1 Adjusted Perplexity

In this subsection, I evaluate my personalized models, along with non-personalized background models, using adjusted perplexity, for differing amounts of text from the user.

In Figure 3.1, I show the adjusted perplexity for models that involve the neural_background model. None of the models are the best performing model for every amount of text from the user. For less than 10k tokens of user-specific training data, neural_background_primed_user achieves the best performance, but above 10k tokens, the models that are interpolated with the user-only models start to perform better. For

<Neural/n-gram>_background Neural or n-gram language model trained on *BACKGROUND*
<Neural/n-gram>_user Neural or n-gram language model trained on the user's text
<Neural/n-gram>_background+ **<neural/n-gram>_user** <Neural/n-gram>_background interpolated with a neural or n-gram language model trained on the user's text
Neural_background+n-gram_background Neural_background interpolated with n-gram_background
Neural_background_primed_user Neural_background, primed on the user's text
Neural_background_continue_training_user Neural_background, with training continued on the user's text

Table 3.2: Description of language models.

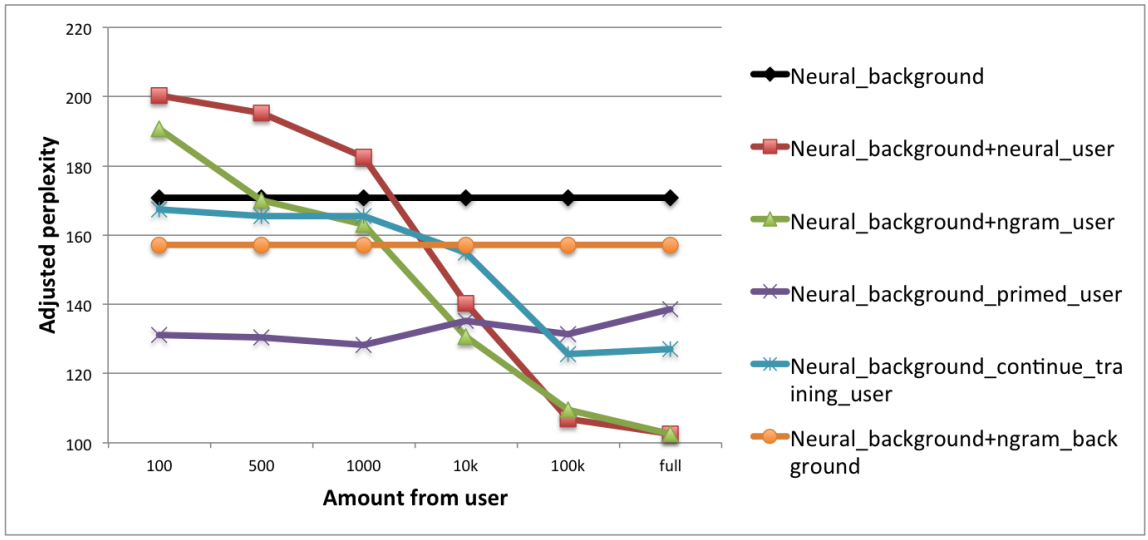


Figure 3.1: Adjusted perplexity for personalized and non-personalized models that include the neural_background model. (Lower is better.)

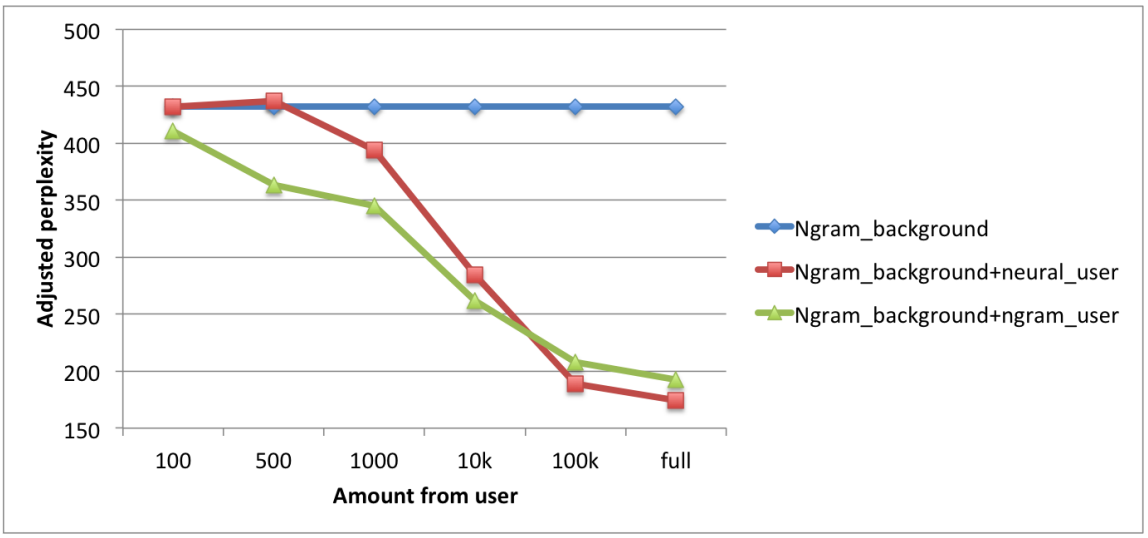


Figure 3.2: Adjusted perplexity for personalized and non-personalized models that include the n-gram_background model.

100k tokens and more, neural_background_continue_training starts to outperform neural_background_primed_user but still does not outperform the interpolated models. I also included neural_background interpolated with n-gram_background for comparison, which outperforms neural_background by itself. Even though n-gram_background, n-gram_user, and neural_user all perform poorly by themselves, they are all able to improve the performance of neural_background when interpolated with it.[1] Neural_background_primed_user achieves a similar adjusted perplexity, regardless of the amount of text from the user. This might be due to how many steps in the past that the state can remember, i.e., the state may largely be unaffected by the more distant tokens used for priming. This phenomenon was explored by Khandelwal et al. (2018).

In Figure 3.2, I show the adjusted perplexity for models that involve the n-gram_background model. We see that n-gram_background interpolated with either of the user-only models (n-gram_user and neural_user) outperforms n-gram_background by itself when at least 1000 tokens are available. n-gram_background interpolated with n-gram_user outperforms n-gram_background for all amounts of text from the user, and outperforms n-gram_background interpolated with neural_user when less than 100k tokens from the

[1]These three models are not shown due to their large adjusted perplexities. n-gram_user does not outperform neural_background regardless of the amount of text from the user.

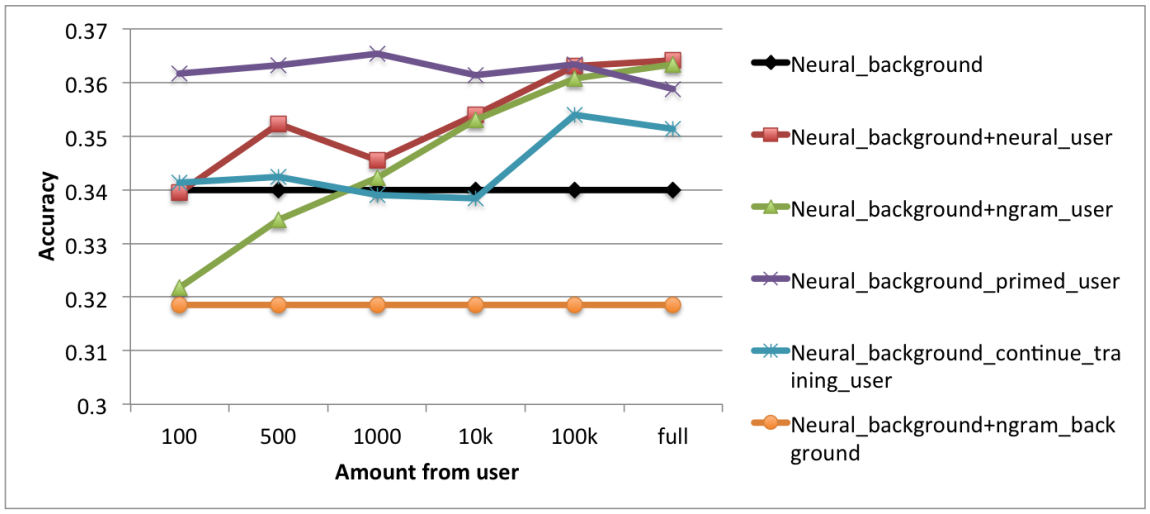


Figure 3.3: Accuracy@3 for personalized and non-personalized models that include the neural_background model. (Higher is better.)

user are used. The performance of neural_background interpolated with neural_user being better with larger amounts of text is most likely a result of neural models requiring more text to train. This is interesting since n-gram_user outperforms neural_user for all amounts of text.[2]

Evaluating with adjusted perplexity showed that a background model interpolated with a user-only model performs well when larger amounts of user-specific text are available. In particular, neural_background interpolated with either user-only model outperforms all other models when using 10k tokens or more of the user's text. The priming method achieves the best performance when given fewer than 10k tokens from the user.

3.6.2 Accuracy@3

In this subsection, I evaluate my personalized models, and non-personalized background models, using accuracy@3 for differing amounts of text from the user.

In Figure 3.3, I show the accuracy@3 for models that involve neural_background. One of the main differences from adjusted perplexity for the neural models is that now neural_background_primed_user outperforms all models, for each amount of text from the user, except for the case of the full amount of text available. Moreover, neu-

[2]User-only models are not shown in the graph due to their poor performance when using less than 10k tokens for training.

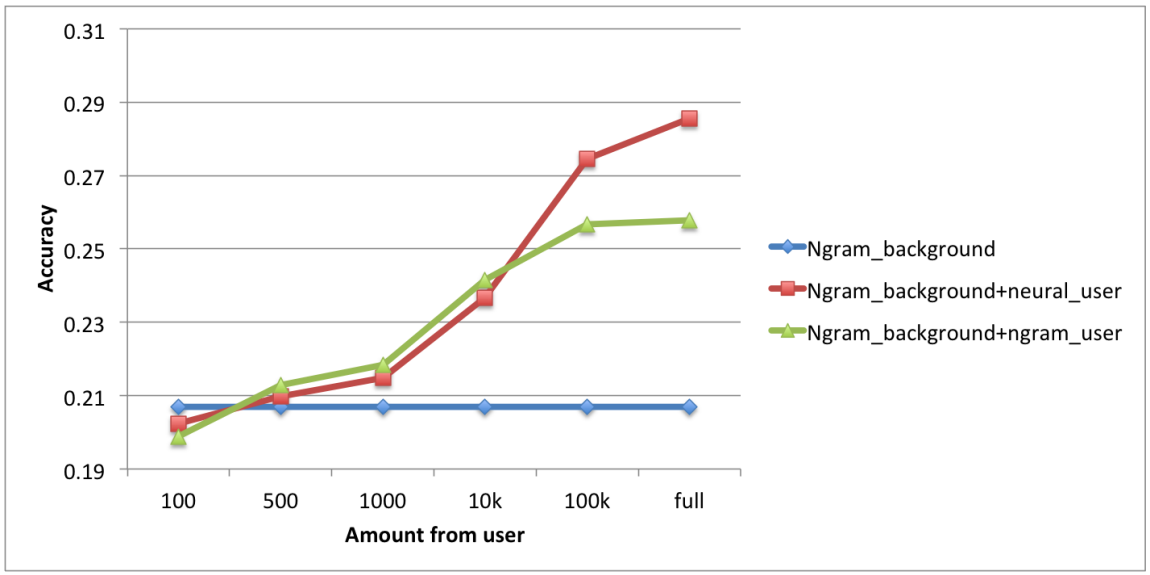


Figure 3.4: Accuracy@3 for personalized and non-personalized models that include the n-gram_background model.

ral_background_primed_user with 1000 tokens from the user outperforms all other models, including those that use the full amount of user-specific text. Neural_background interpolated with neural_user now outperforms neural_background interpolated with n-gram_user. The improvement with the use of neural_user over the use of n-gram_user for only accuracy@3, and not adjusted perplexity, might be because of accuracy@3 being a more relative metric than adjusted perplexity. For example, a model needs to assign high probability for the correct type to perform well under adjusted perplexity (which a standard neural model does not do with a small amount of training text), but with accuracy@3, a model only needs to generate a probability for the correct type that is greater than other types in the model's vocabulary. This figure also shows that interpolating the two background models no longer outperforms neural_background by itself. Neural_background_continue_training_user outperforms neural_background when at least 100k tokens from the user are available, as opposed to the adjusted perplexity metric, which showed that only 10k tokens are needed for neural_background_continue_training_user to outperform neural_background.

In Figure 3.4, I show that n-gram_background interpolated with either user-only model outperforms n-gram_background by itself when the user-only model is trained on at least

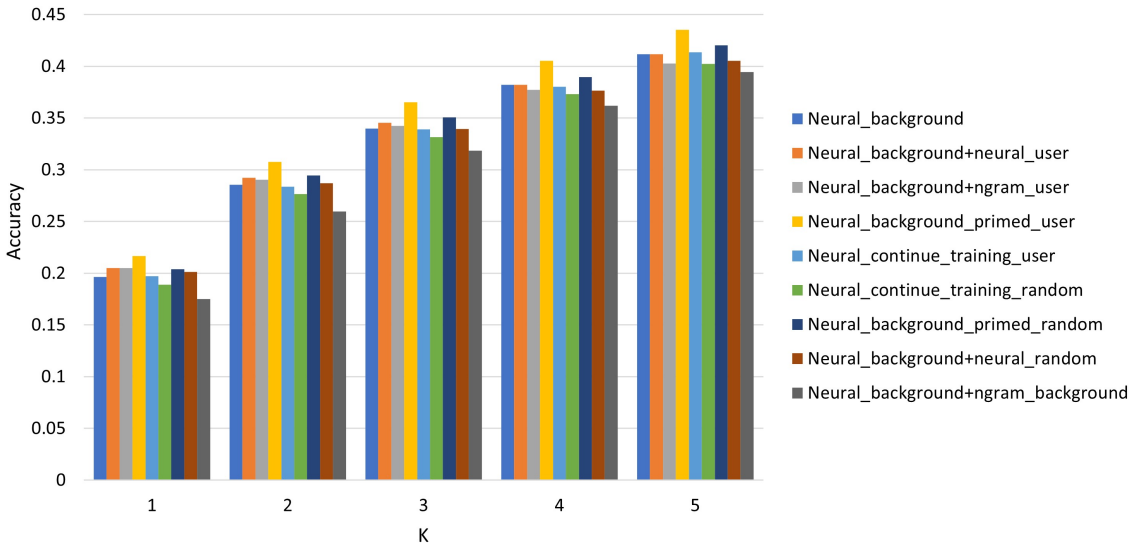


Figure 3.5: Accuracy@k for personalized models that were tuned on 1000 tokens and non-personalized models that include the neural_background model. (Higher is better.)

500 tokens. There is not a large difference between the two interpolated models until 100k tokens are used, when n-gram_background interpolated with neural_user starts to largely outperform n-gram_background interpolated with n-gram user. This supports the earlier observation of neural models requiring more text for training to outperform n-gram models. This evaluation with accuracy@3 showed that priming performs the best, or similar to the best model, for all amounts of user-specific text considered. Moreover, priming with 1000 tokens performed the best of all models considered, even those using orders of magnitude more user-specific training data.

3.6.3 Accuracy@k

In this subsection, I evaluate personalized and non-personalized models that involve the use of neural_background, using the accuracy@k metric.

In Figure 3.5, I show accuracy@k for models that involve neural_background. The personalized models were tuned on only 1000 tokens. It shows that neural_background_primed_user always outperforms all other models for each value of k. To evaluate the importance of using text from the original user, I apply the same priming technique on an identical background model but use text from *RANDOM*. This model is labeled as neural_primed_random

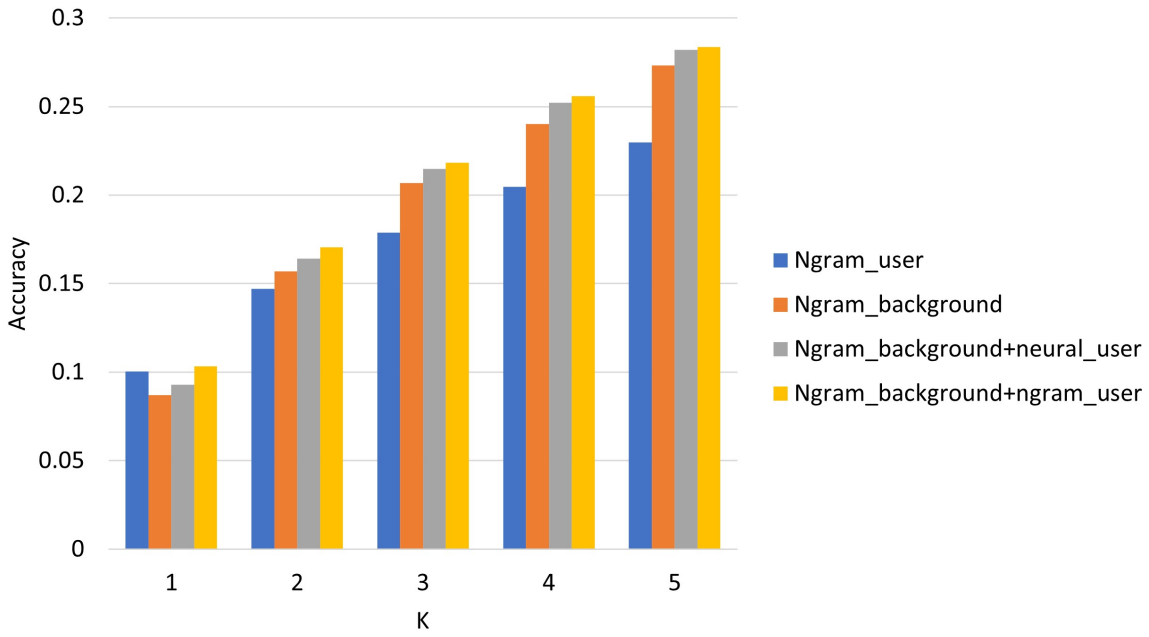


Figure 3.6: Accuracy@k for personalized models that were tuned on 1000 tokens and non-personalized models that include n-gram-based models.

and achieves a poorer performance than the interpolated models for values of k less than 3, but outperforms them for values of k greater than or equal to 3. This might be demonstrating neural_primed_random's ability to capture more generic information and ranking the correct word in the top 3 to 5 positions, but not focused enough on user-specific text to rank it in the top 2 positions. This further suggests the need to consider personalization.

In Figure 3.6, I show accuracy@k for models that include n-gram-based models for values of k from 1 to 5 and using 1000 tokens from the user. It shows that ngram_background+ngram_user achieves the highest performance for all values of k. Interestingly, ngram_user almost achieves the highest score for accuracy@1 but performs relatively poorly for all other values of k. Although ngram_user performs poorly, it assists in the interpolated model ngram_background+ngram_user. Both interpolated models consistently outperform ngram_background. Due to models that use neural_background greatly outperforming models that use n-gram_background, I do not further consider models that use n-gram_background.

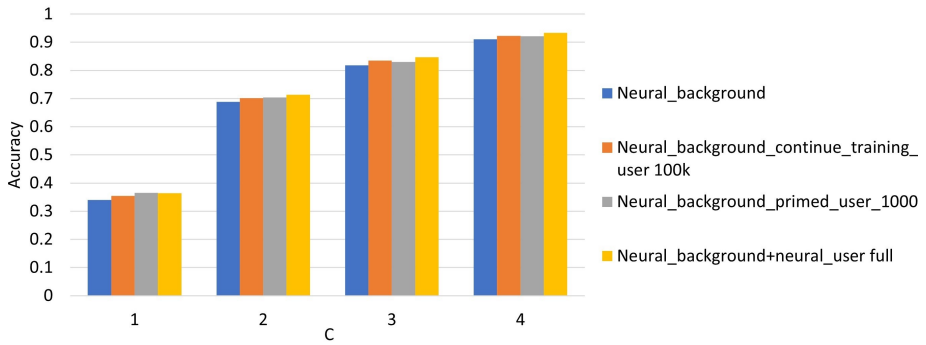


Figure 3.7: Accuracy@3 given c keystrokes for selected models.

3.6.4 Accuracy@3 Given c Keystrokes

Figure 3.7 shows accuracy@3 given c keystrokes for some of the best performing models, along with neural_background. For each approach to personalization, I consider the amount of text from the user that gave its best performance in terms of accuracy@k. Specifically I use 100k tokens for neural_background_continue_training, the full amount of user-level text for neural_background interpolated with neural_user, and 1000 tokens for the priming method.

Given only one character, all of the models achieve approximately 70% accuracy@3. Neural_background_primed_user achieves the best performance when no characters are available to the models, but neural_background interpolated with neural_user achieves the best performance when there is at least one character available. This may be due to the introduction of the user-only model's vocabulary, resulting in non-zero probabilities for more words that were used by the user.

Evaluating with accuracy@3 given c keystrokes, we see that each of the approaches to personalization outperforms the background model, for all values of c, and that the interpolated model performs best when at least one keystroke is given. Nevertheless, the priming approach achieves similar accuracy to the best approach for each value of c, while requiring much less user-specific training data.

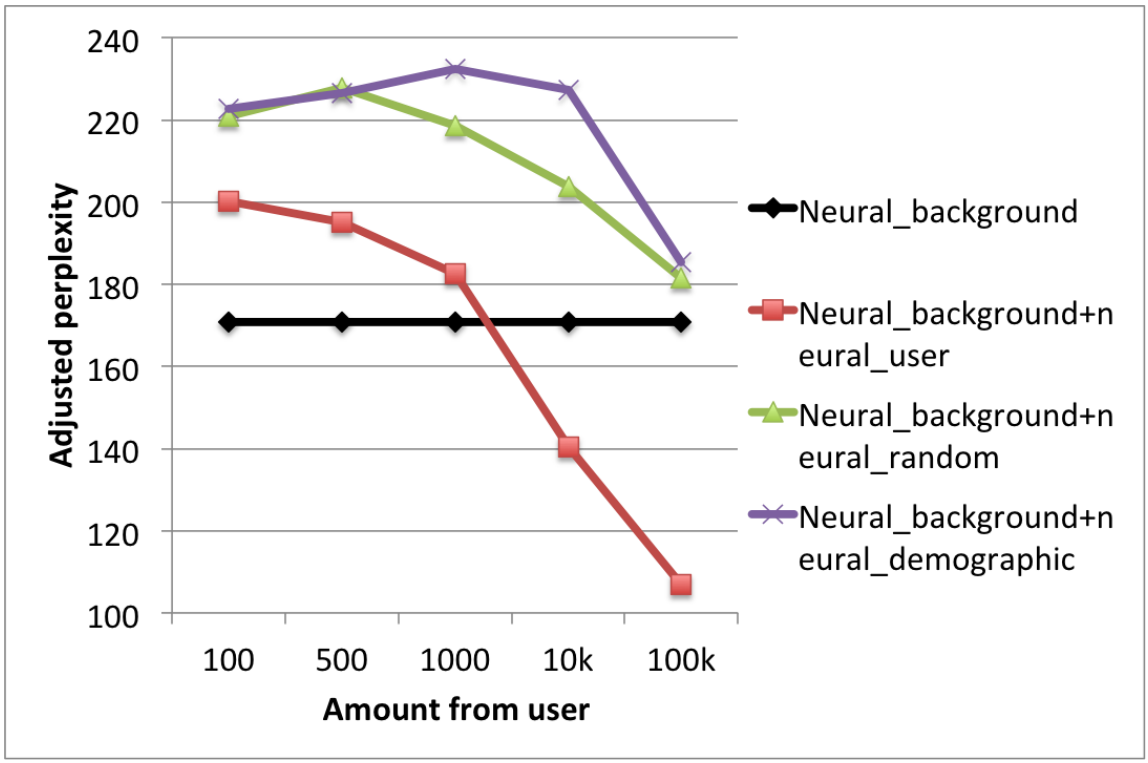


Figure 3.8: Adjusted perplexity for neural_background and interpolated models.

3.6.5 Comparing to RANDOM and DEMOGRAPHIC

In this section, I compare the approaches to personalizing language models with variations of these approaches in which text from the user (i.e., *USER-TRAIN*) is replaced with text from *DEMOGRAPHIC* or *RANDOM*. I carry out these experiments with *DEMOGRAPHIC* and *RANDOM* to determine whether language model adaptation based on demographic characteristics is as effective as personalization, and whether the improvements in model performance with personalization are not simply a result of more text being available to the models.

In Figures 3.8 and 3.9, I show the adjusted perplexity and accuracy@3 for models based on interpolation.[3] Neural_background interpolated with neural_user always outperforms neural_background interpolated with random and neural_background interpolated with neural_demographic.[4]

[3]For experiments in this subsection, the largest corpus size I consider is 100k tokens. Using all text available for *RANDOM* would give a much larger corpus than for any individual user in *USER-TRAIN*, and for *DEMOGRAPHIC* would give corpora of widely varying sizes.

[4]Neural_demographic was trained using a batch size of 45, which is the value that was used to train

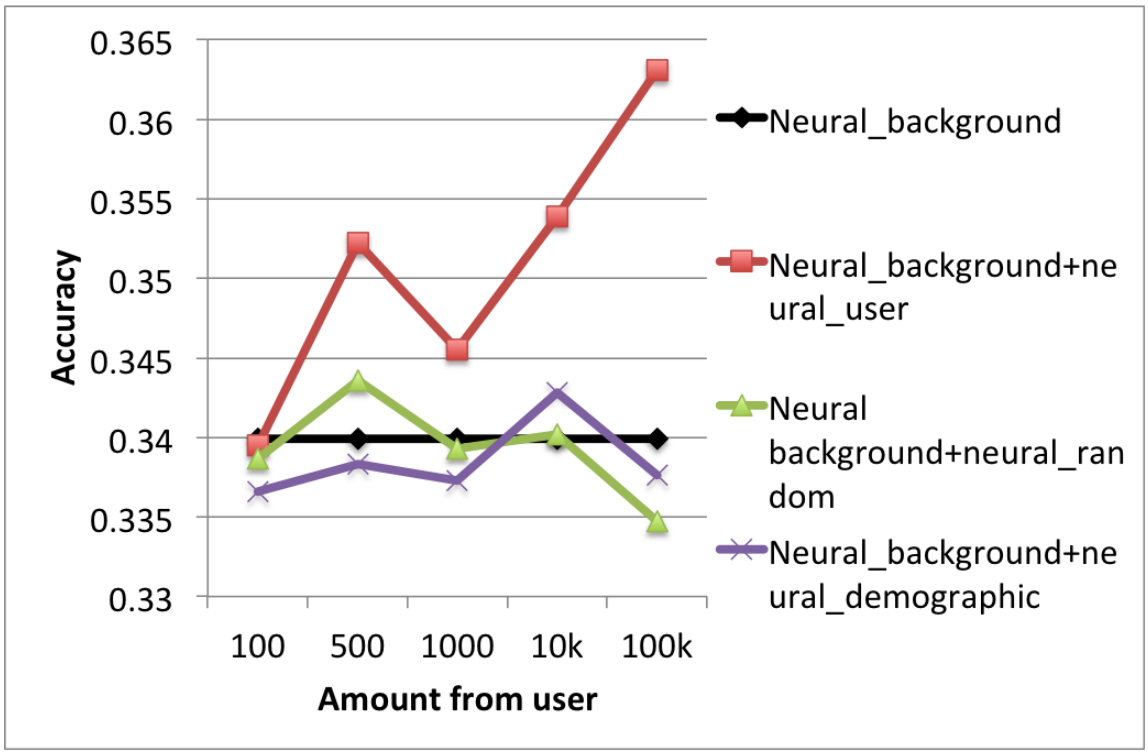


Figure 3.9: Accuracy@3 for neural_background and interpolated models.

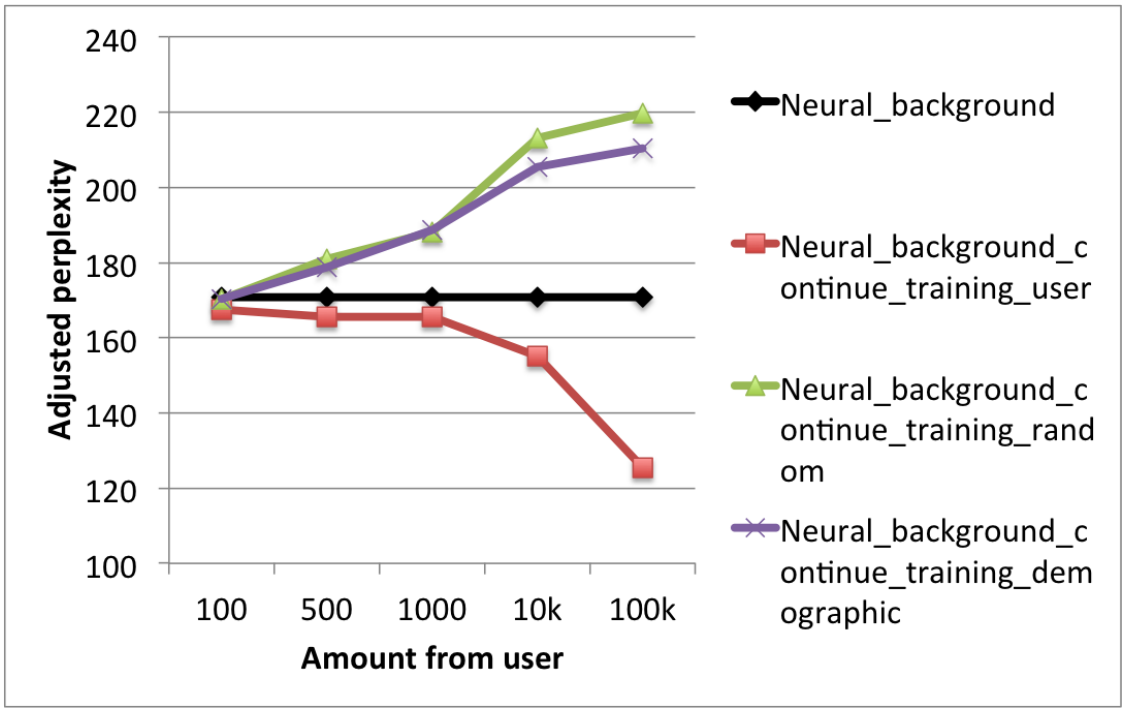


Figure 3.10: Adjusted perplexity for neural_background and models that continue training on a second corpus.

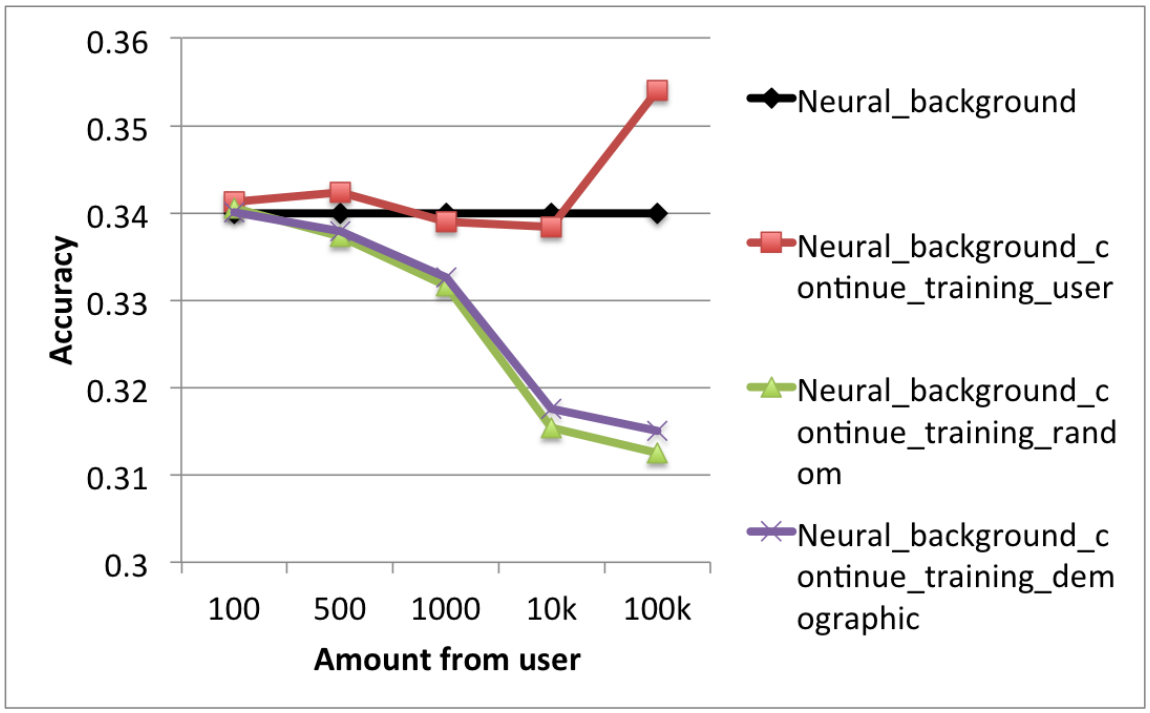


Figure 3.11: Accuracy@3 for neural_background and models that continue training on a second corpus.

In Figures 3.10 and 3.11, I show the performance of models that continue training on a second corpus. Figure 3.10 shows that continuing to train on user-specific text always outperforms neural_background by itself. Both Figures 3.10 and 3.11 show that continuing to train on user-specific text always outperforms continuing to train on *RANDOM* or *DEMOGRAPHIC*. There is an unexpected behaviour (models achieving poorer results when further trained on text) from the models that continue training on *RANDOM* and *DEMOGRAPHIC*, which might be due to how these corpora were generated. To build *RANDOM* and *DEMOGRAPHIC*, I randomly select sentences, meaning that consecutive sentences in the corpora are most likely not actually consecutive sentences in blog posts. My models train across sentences using sequences of 30 tokens, and therefore, the words seen in a sequence can be "out of place" when trying to predict the target word. This effect was not present in the interpolated models. This might be because the continue training model starts with the neural_background model — which was trained on sentences that were in the correct order — and then continues training on sentences that are not in the

neural_background

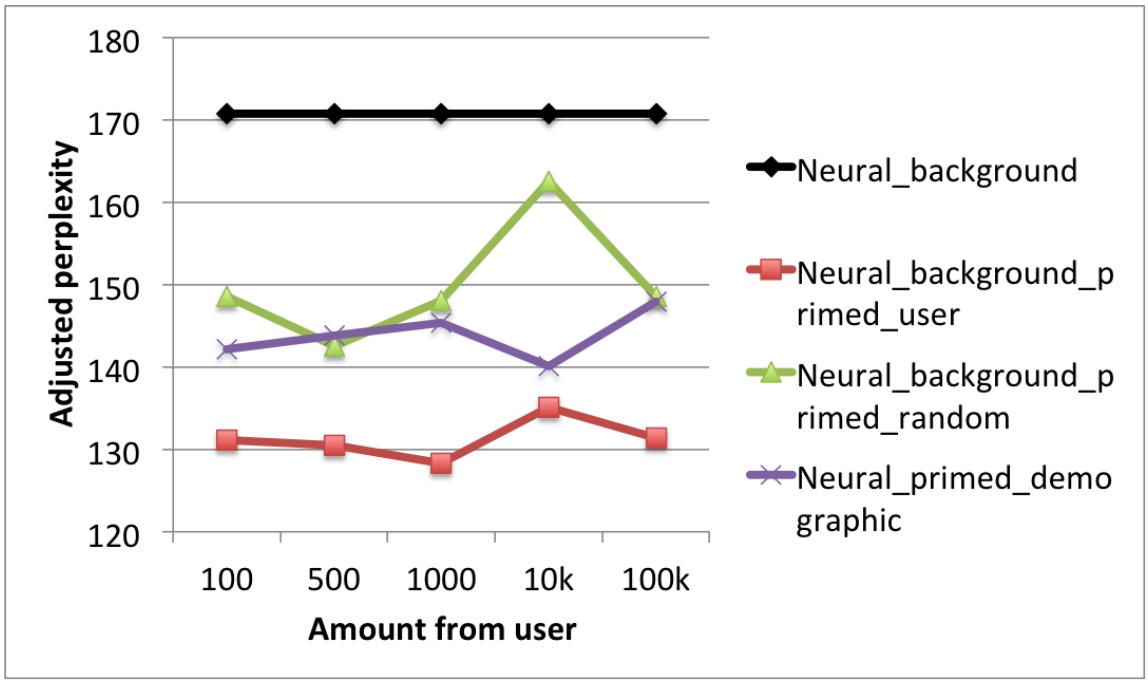


Figure 3.12: Adjusted perplexity for neural_background and neural_background primed on a second corpus.

correct order. On the other hand, the language models trained on the second corpus for the interpolated models only see sentences that are out of order.

In Figures 3.12 and 3.13, I show that all primed models outperform the unprimed neural_background model, which suggests that priming on any in-domain text can improve performance. Neural_background primed on user-specific text outperforms all other models, for all amounts of text considered. Neural_background primed on *DEMOGRAPHIC* outperforms neural_background primed on *RANDOM* — showing that model adaptation based on demographics is indeed useful, but less effective than personalization. The issue that was present in models that continue training in Figure 3.11 for *RANDOM* and *DEMOGRAPHIC* does not appear to affect primed models, which might be due to the state of the LSTM being less affected by words further away from the target word — making words outside the current sentence less likely to affect the state.

These experiments show that language models that are adapted using text from a specific user — i.e., personalized models — outperform models that are adapted based on demographic factors, and models that simply use additional in-domain text for adapta-

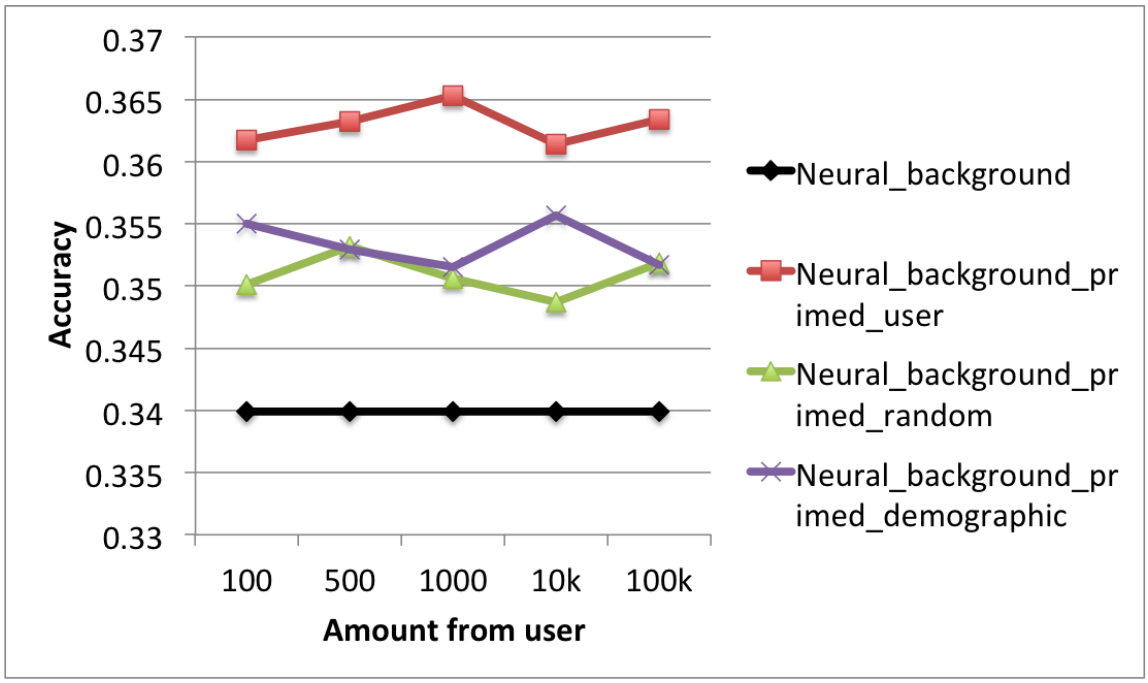


Figure 3.13: Accuracy@3 for neural_background and neural_background primed on a second corpus.

tion, for all amounts of text used for model adaptation, and for the two evaluation metrics considered.

3.7 Summary

In this Chapter, I showed that a background language model can be improved through personalization for a specific user, while requiring relatively little text from that user. I considered three different personalization techniques, and three evaluation metrics. I showed that for adjusted perplexity and accuracy@3, each personalization technique can improve over the background model, with interpolated models performing better when a relatively large amount of text is available from the user, while priming performs better when a relatively small amount of text from the user is available. I further showed that priming outperforms all other models, even those which use orders of magnitude more user-level text for personalization, for accuracy@3. For accuracy@3 given c keystrokes, I showed that the best performing models get a large increase in performance — from

approximately 0.35 to 0.7 — when at least one keystroke is given. Finally, I showed the importance of personalization over model adaptation based on demographic factors. For each personalization technique, using text from a specific user for model adaptation outperforms using text from users with similar demographic characteristics.

Chapter 4

Authorship Verification

In this chapter, I discuss my models for authorship verification, which involves determining if a person is the author of a given document. Specifically, I focus on text from social media users and use potentially malicious text as negative instances, such as hate speech, email spam, and erotica. This work is an extended version of King et al. (2018) and King and Cook (2020a).

4.1 Framework

In this work, I propose and evaluate two different types of methods for authorship verification. The first method involves representing a document by embedding it, and then using a one-class SVM for classification. The second method involves personalizing a language model by tuning it on text from an author, and comparing its probabilities to a non-personalized language model. I evaluate my models while considering the amount of text available for training and the size of the documents used for classification. I show that my proposed personalized language model-based methods outperform embedding-based methods, while requiring orders of magnitude fewer tokens from the user.

I frame the authorship verification task as predicting whether an unknown document belongs to a given author or not, while crucially only observing documents from that author during training, i.e., the model does not see negative instances during training. This resembles the real-world scenario where we do not know in advance what kinds of malicious

documents an unauthorized user of a social media account would attempt to post. The lack of negative instances during training means that standard supervised approaches to binary document classification are not applicable.

4.2 Dataset and Evaluation

Here, I discuss the process that I use to create my dataset, and the evaluation measures that are used to evaluate models.

4.2.1 Dataset

In this subsection, I describe the sources of positive and negative instances that were used to create my dataset, and then the structure of the dataset. The positive instances are texts from a specific author and the negative instances are texts from one of three malicious text types that are discussed in the following subsection.

Sources of Text

The positive instances in my dataset consist of blog posts from a corpus containing $19,320$ authors (Schler et al., 2006) i.e., the same corpus used to construct the dataset for experiments in Chapter 3. I select all authors who have at least 300 posts with at least 100 tokens in them, giving me 103 authors for experiments. For each of these 103 authors, I ignore their 10% smallest and 10% largest documents to avoid outlier documents.

The texts belonging to the remaining authors are combined to generate a background corpus for training my language models, which consists of approximately $143M$ tokens. The text from any one author is limited to 30k tokens to avoid text from any one author strongly biasing the corpus to represent their text. I replace each word in the background corpus that has a frequency less than 10 with a special token (UNK). This is to reduce the cost of training language models.

The negative instances were gathered from three different sources, specifically, erotic stories, hate speech, and email spam. I use several types of malicious documents to avoid the task being framed as document classification for a specific (malicious) text type. For

example, if I used only hate speech, then the task could be approached as identifying hate speech, as opposed to the broader authorship verification task that I consider.

All erotic stories are gathered from `textfiles.com/sex/EROTICA`. I arbitrarily selected all plaintext documents with titles starting with "A" or "B", which resulted in a relatively large amount of text. I removed lines that contain metadata from each document. I then selected all documents that contain at least 200 tokens, giving me 1463 documents.

I gathered hate speech documents from a white supremacy forum that were used in the dataset of de Gibert et al. (2018). The dataset originally contained individual sentences from the documents with information about which sentences are from the same document and the order in which they appear. I reconstructed the documents using this information and selected all documents that contain more than 50 tokens. This resulted in 172 documents in total.

I gathered email spam from the Enron-Spam dataset (Metsis et al., 2006). I removed lines containing the text "subject:" and removed paragraph boundaries. I then selected all documents containing at least 200 tokens, resulting in 6230 documents.

All documents — positive and negative instances — were casefolded and tokenized using the Stanford Core NLP toolkit (Manning et al., 2014), except for the hate speech texts, which were pre-tokenized. All training and testing documents are prepended with a start-of-sentence token and appended with an end-of-sentence token.

Dataset Structure

I split my dataset into a development set (*DEV*) and a testing set (*TEST*). I randomly select 10 authors from the 103 blog post authors for *DEV*, which is used for preliminary experiments. I assign the remaining 93 authors to *TEST*, which is used for evaluation. The authors in *TEST* include 48 males and 45 females, with an average age of 27 years, ranging from 14 to 48 years old.

The following design is used for both *DEV* and *TEST*. For each author, I create user-specific training (*DEV_train* and *TEST_train*) and testing (*DEV_test* and *TEST_test*) datasets. For each author, I randomly select 45 of their documents for testing, which make up the positive instances in *DEV_test* and *TEST_test*. The remaining documents

for each author are put into *DEV_train* and *TEST_train*. For each author, I then select 15 erotica, 15 hate speech, and 15 spam documents as negative test instances, and add them to the author-specific testing sets (i.e., *DEV_test* and *TEST_test*). A given malicious document is never included in both *DEV* and *TEST*. This setup assigns each author 45 malicious documents that do not belong to them. I repeat this 5 times, to create 5 different test sets for each author. All test documents (positive and negative instances) are selected without replacement for an individual author across all 5 test sets. For example, if a document is selected for the first test set then it is removed from the pool of possible instances for other test sets. As such, a given document will only occur in *TEST_test* at most once for a given author.

I limit negative instances to approximately the first 1000 tokens. This reduces computational cost. Moreover, I control the number of tokens used from test documents in my experiments, and none of the authorship verification approaches considered use document length as a feature.

In *TEST*, the average amount of text from an author is approximately $200k$ tokens, with a minimum and maximum of $97k$ and $526k$ tokens, respectively.

4.2.2 Evaluation

I evaluate my models on the 93 authors from *TEST* over 5 iterations. In each iteration, I select documents for training/tuning by concatenating documents from *TEST_train* until I have at least x tokens, where x is a parameter that controls the amount of text available for training. I then give the model y tokens of running text from a test document from *TEST_test* for classification, where y is a parameter that controls the amount of text used from a test document. Limiting the amount of text from training documents simulates having only a small amount of text from the user available for building the model; limiting the amount of text from test documents simulates documents of various lengths, e.g., microblogs vs longer documents, such as blogs.

I evaluate my models using accuracy and F_1 score. My test sets contain a 50/50 split of positive and negative instances, which makes accuracy an appropriate metric.

4.3 Methods

In this section, I describe my two methods for authorship verification based on embeddings and language models.

4.3.1 Embeddings

In this method, I represent documents as embeddings, using either word2vec (Mikolov et al., 2013) or DistilBERT (Sanh et al., 2019), and then use a one-class SVM for classification. For word2vec, I train skipgram (Mikolov et al., 2013) on a snapshot of English Wikipedia with an embedding size of 300 and a window size of ± 8, and then represent a document as the average of its word embeddings.

DistilBERT(Sanh et al., 2019) is a lighter-weight version of the BERT transformer model with 6 layers, an embedding size of 768, and 12 attention heads. These are the default parameters for the DistilBERT implementation from Wolf et al. (2019), which I use. I embed a document by using the document as input to the model, which generates an embedding for the document as part of its output layer.

I represent training documents using these approaches to generate document representations — i.e., either word2vec or DistilBERT — and then train a one-class SVM. Given a test document, I use the same approach to embed it, and then input it into the trained one-class SVM to classify the document as belonging to the author or not. I use the one-class SVM implementation from scikit-learn.[1] I use the default parameters for this model, which includes a radial basis function kernel.[2] I refer to these approaches which use a one-class SVM along with document representations based on either word2vec or DistilBERT as *W2V* and *BERT*, respectively. King et al. (2018) showed that *W2V* outperformed the previous state-of-the-art approach of Jankowska et al. (2014) from Section 2.5.2.

4.3.2 Language Models

In this method, I require two language models. The first language model, referred to as the background language model, is trained on a background corpus of blog posts (described

[1] `https://scikit-learn.org`

[2] These parameter settings could be tuned on *DEV*. I leave this for future work.

in Section 4.2.1). The second language model uses the priming personalization technique discussed in Section 3.5.3. To recap, the personalized language model is generated by copying the first background language model, and then exposing it to text from the author's training documents to modify the LSTM's state without altering the hidden layer of the network that is tuned during standard training. My language models consist of an LSTM with a single hidden layer with 1024 units, an embedding size of 128, and are trained for 1 epoch with a batch size of 45.

Given a test document, each language model — i.e., the background and personalized language model — outputs a probability for every target token in the document. I score each model by counting the number of target tokens that the model assigns a higher probability than the other language model. For example, if language model A assigns a probability of 0.7 to the target token and language model B assigns a probability of 0.1 to the target token, then language model A is awarded the point for this token in the document. This is performed for each token in the document. I classify the document as belonging to the author (i.e., a positive instance) if the personalized language model scored higher than the non-personalized background language model. I classify the document as not belonging to the author (i.e., a negative instance) if the background model achieved a higher score. Ties are labeled as not belonging to the author. In preliminary experiments on *DEV*, I considered alternative approaches, including comparing the perplexity of the two language models, and thresholds for differences in probability. None of these approaches performed as well as my proposed approach. I refer to the proposed method as *LM* from hereon.

In preliminary experiments on *DEV*, I found *LM* performed well on shorter test documents, but performed poorly as the length of the test documents increased. I believe this was due to the fact that the state of the LSTM changes fairly quickly as a test document is processed. As such, for longer documents, the state of the background language model becomes similar to the state of the personalized language model, and the usefulness of having a non-personalized model is lost. To address this, I sentence-tokenized the test documents using NLTK's sentence tokenizer (Bird et al., 2009), and reinitialized the personalized language model's state to its original primed state at the beginning of each sentence. Each sentence

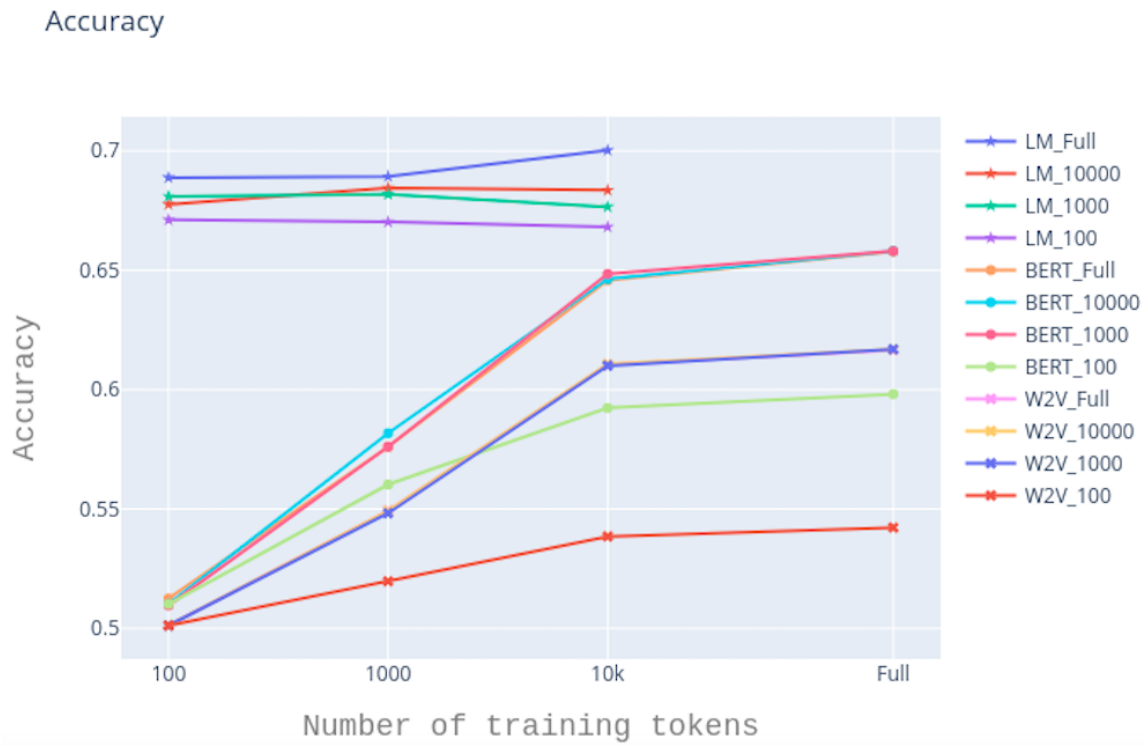


Figure 4.1: Accuracy for each method, using different amounts of training text, and different test document sizes. The number appended to each model's name in the legend indicates the cut-off length for the number of tokens used from the test documents; i.e., *BERT_100* indicates *BERT* with test documents cut-off after the first 100 tokens. (W2V_1000, W2V_10000, and W2V_Full achieve similar accuracies, and therefore, only W2V_1000 can be seen.)

had the beginning-of-sentence and end-of-sentence tokens appended to the start and end of the sentence, respectively. I found that this sentence-level re-initialization always led to improvements, and so I only report results for this approach.

4.4 Experimental Results

In this section, I evaluate my proposed models using accuracy and F_1 score. I consider different amounts of user-specific training text, to simulate having varying amounts of user-specific data available. I further consider different cut-off lengths for test documents, to simulate different test document sizes.

Figure 4.1 shows the accuracy of my models when using different amounts of user-specific training data, and differing cut-offs for test document sizes. I do not report results for *LM*

using all available user-specific training data because these experiments are computationally expensive, and preliminary experiments indicated that this approach performed very well with only modest amounts of training data.

Results in terms of accuracy are shown in Figure 4.1. *LM* outperforms both embedding-based models — i.e., *W2V* and *BERT* — regardless of the amount of training text or test document size. Remarkably, *LM* achieves an accuracy of 0.69 with only 100 tokens of user-specific training text when using the full test documents. The embedding-based models perform relatively poorly when only a small amount of user-specific training data is available, but perform better as more training text is used. For every amount of training data, and test document length, considered, *BERT* always outperforms W2V. The highest accuracies achieved by *BERT* and *W2V* are 0.66 and 0.62, respectively. Unlike the embedding-based models, *LM* does not always perform better when more training text is available, which could be because the state of the LSTM does not retain much information from tokens that are far away (Khandelwal et al., 2018).

Figure 4.1 also shows that the embedding-based models achieve close to their highest values when using only 1000 tokens from the test documents, and do not perform substantially better on longer test documents. This could be due to the construction of the dataset, where malicious documents are truncated to approximately 1000 tokens. I do not apply the same document size limitation to documents from the user. *LM* generally performs better on larger test documents, and achieves its best accuracy of 0.70 using full test documents and $10k$ tokens of user-specific training text.

Figure 4.2 shows the F_1 score of the models for different amounts of training data and test document sizes. The findings are overall similar to those in Figure 4.1, with *LM* outperforming the embedding-based models, and achieving a highest F_1 score of 0.74, and an F_1 score of 0.73 when trained on only 100 tokens of user-specific text. The best F_1 score of an embedding-based model is 0.57. Interestingly, here the performance of *LM* is not overly affected by the test document length. One difference here is that *BERT* no longer consistently outperforms *W2V* for the same amount of training text and test document size.

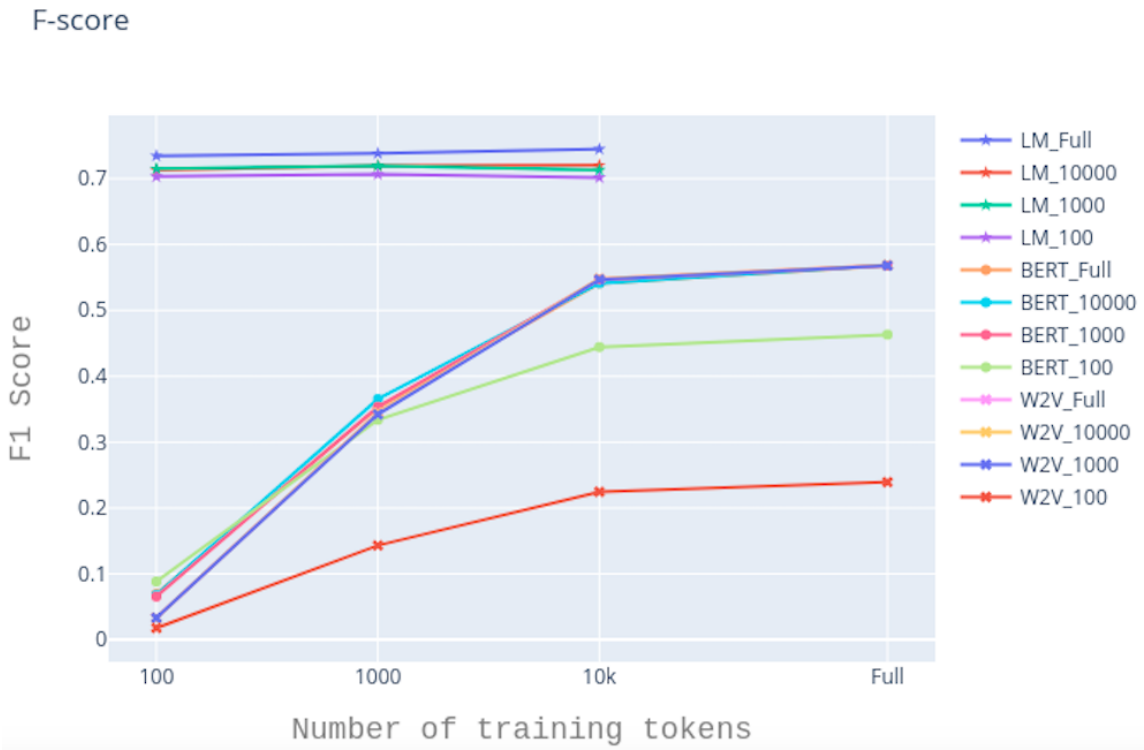


Figure 4.2: F_1 score for each method, using different amounts of training text, and different test document sizes.

4.5 Summary

In this chapter, I explored authorship verification with a focus on preventing malicious content from being posted to a social media account. I showed that the use of personalized language models can outperform the previous state-of-the-art embedding-based models on an authorship verification task while requiring far less text from the user. This ability to perform well with relatively low amounts of text is especially important for users that do not possess a large amount of text. These models assume that all text that is associated with an author was written by that author. Therefore, any text that is being quoted or reposted would be considered written by this author. This is a vulnerability of this model and would make this model unsuitable for plagiarism detection.

Chapter 5

Word Sense Disambiguation

In this chapter, I propose a novel dataset for word sense disambiguation (WSD) that enables studying personalized WSD. I also propose and evaluate a variety of personalized WSD methods on this dataset.

5.1 Framework

Authors of text tend to predominantly use a single sense for a given lemma, and this predominant sense can differ among authors. This might not be captured with an author-agnostic WSD model that was trained on multiple authors. Furthermore, my work finds that WordNet's first senses, the predominant senses of my dataset's genre, and the predominant senses of an author can all be different. Therefore, author-agnostic models could perform well over an entire dataset, but poorly on individual authors. Ideally, each author would have access to a personalized WSD model, which is a model that is tailored toward that individual.

In this chapter, I show that personalizing a WSD system by tailoring existing state-of-the-art models toward an individual by exploiting the author's sense distributions can improve the performance. Specifically, I evaluate different WSD models, including a state-of-the-art model (*SensEmBERT*) (Scarlini et al., 2020), which I extend with personalization techniques to achieve my best scores. Furthermore, my work shows that the use of author-specific sense distributions outperforms the use of genre-specific sense distributions,

further demonstrating the benefits of personalization. The evaluation is performed using my proposed dataset which contains 1586 sense-annotated instances for 11 lemmas across 36 authors. My evaluation includes metrics that focus on the performance of models with respect to the authors that they perform most poorly on, which highlights the potential gain for individual authors and the fairness of the models.

In this first work on personalized WSD, I do not automatically learn the sense distributions of an author, but instead, I use the author's true sense distributions to demonstrate the importance of learning author-level sense distributions when considering personalized WSD.

5.2 Data Statement

In this section, I discuss the properties of my dataset, while considering the proposed schema from Bender and Friedman (2018).

5.2.1 Data selection

I collect all text from blogs of the top 50 authors that contain the most tokens from the corpus that was originally presented in Schler et al. (2006), again the same corpus used to construct the datasets for experiments in Chapters 3 and 4. The original corpus consists of English blog posts from $19,320$ authors. Due to the nature of the corpus' creation, there were some authors who possess the same text as other authors in their entirety, and therefore, I do not consider duplicated authors in the top 50. I consider the top 50 authors to ensure that each author possesses enough text to allow the ability to study the potential benefits of using text from the author for personalizing a WSD model. For selecting lemmas, I first consider all 20 nouns from Gella et al. (2014). I consider the top 10 authors — authors that have at least the 10th highest frequency of a lemma — to ensure that there is text from multiple authors for each lemma to study the effects of personalization on a per-lemma basis. From this group of lemmas, I retain all lemmas that have been used 20 or more times by the author who has used this lemma the 10th most frequently. I selected the cutoff of 20, because there is a possibility that all instances from

an author will not be usable due to them not being a noun or not representing a sense from my chosen sense inventory. The group of lemmas that I include moving forward will be referred to as *SHORT_LIST*.

For each lemma in *SHORT_LIST*, I randomly sample approximately 10 sentences that contain the lemma from 10 authors and manually assign the lemma a sense.[1] I then use this annotated subset to perform two different analyses that focus on quantifying the diversity of the senses for a lemma. The first analysis involves calculating the predominant sense of each lemma for each author and then finding the most frequent sense among the author-level predominant senses, which I call the grand sense. I then calculate the percent of author-level predominant senses for a given lemma that are not the grand sense. The second analysis calculates the number of assigned senses that are not the grand sense. Both types of analysis assist in showing which lemmas have senses that vary among authors, and therefore, they might benefit from a personalized model which is the main focus of this work. Lemmas that score low on these metrics could be ideal for models that predict the predominant sense, but would most likely not benefit from an author-level model. I originally wanted the top 10 lemmas that scored the highest when comparing predominant senses with the grand sense, but I received a tie for the lemmas that scored 9th, 10th, and 11th. I remove all lemmas from *SHORT_LIST* that scored less than 0.22 on the percent of predominant senses that are not the grand sense, which was the score for the lemmas with the 9th, 10th, and 11th highest values. Table 5.1 shows my final 11 lemmas with their frequency for the top 10 authors, and their two diversity metrics.

Additional preprocessing was applied to the text from authors and the annotated instances, including the replacing of tokens that contain URL identifiers (*www*, *html*, *https*, etc.) with the token *urlLink* and the removal of what appears to be artifacts of text encoding, such as $\backslash xx\backslash xx\backslash xx$.

For each of the 11 lemmas in the dataset, I gather the top 10 authors that use that lemma most frequently and gather all sentences from them that contain that lemma tagged as a noun by a part-of-speech tagger (Qi et al., 2020). This results in 36 authors and a total of 1607 instances across all lemmas. I manually scan through all instances and remove all

[1] I am a native English speaker.

Lemma	10^{th}MF	Predom	Token	# Senses
form	54	0.60	0.69	16
position	40	0.60	0.74	16
degree	27	0.56	0.57	7
sign	77	0.50	0.62	11
track	36	0.44	0.67	12
paper	54	0.44	0.57	7
deal	75	0.40	0.44	9
field	30	0.30	0.43	17
case	97	0.22	0.47	19
charge	36	0.22	0.34	15
rule	43	0.22	0.27	12

Table 5.1: List of lemmas and their frequency for the author who uses the lemma the 10th most frequently (10^{th}MF), and the percent of author-level predominant senses and token senses that are not the grand sense, represented by *Predom* and *Token*, respectively (higher values indicate more diversity). The number of WordNet senses for each lemma under the label *#Senses* is also shown.

instances that were incorrectly tagged as nouns.

5.2.2 Annotation

In this work, I use WordNet (Miller, 1995) as the sense inventory due to its popularity among WSD tasks (Raganato et al., 2017). I show the number of WordNet senses for each lemma in Table 5.1, which ranges from 7 for *degree* to 19 for *case*.

I use Amazon Mechanical Turk — a common crowd-sourcing site — to annotate the instances of each lemma. I group the instances into sets of 5 — known as HITs — and ensure all 5 instances belong to the same lemma. Each instance in a HIT is presented to an annotator, known as a turker, with a piece of text that contains a single target token written in bold for each text. Each instance contains up to 20 tokens before the target token and 20 tokens after the target token, which can cross sentences but does not cross blog posts. This provides the turker with more context than only looking at a single sentence, which can assist their annotations. The turkers are asked to select the sense that best applies to the target token from a list of the WordNet senses for the lemma of the target token or they can select *I cannot assign a sense*. There was space available for feedback for each instance, where turkers can write the reason that they cannot assign a sense or

provide general feedback. It is possible that a token can exhibit multiple senses (Erk et al., 2009), but my dataset will only consider one sense as the ground truth for each instance. Therefore, following Chklovski and Mihalcea (2002) and Pradhan et al. (2007), a turker can only select one sense for any instance. Each HIT was annotated by 10 turkers. An example of the task assigned to turkers is seen in Figure 5.1.

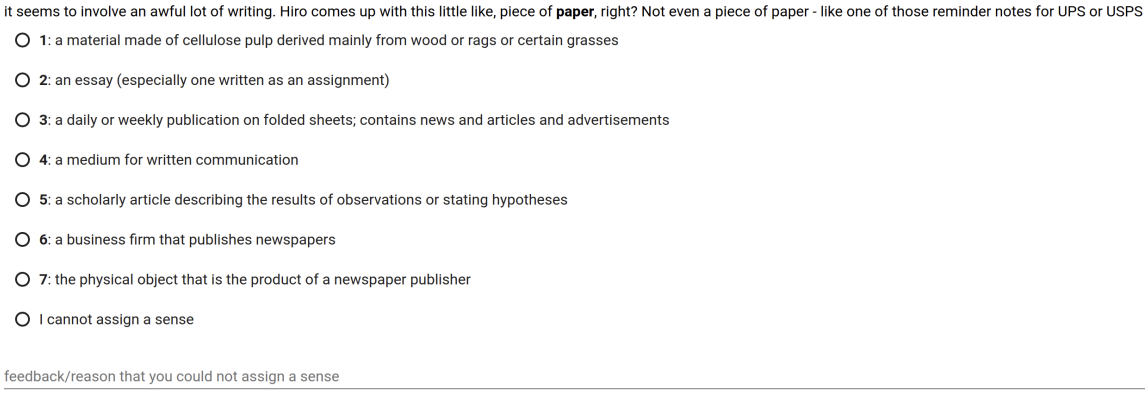


Figure 5.1: Example of the task assigned to turkers.

Annotators

For a turker to be eligible to annotate the HITs, they need to be at least 19 years of age, speak English as a first language, live in Canada or United States, and have a previous HIT acceptance rate of 98%. I paid the turkers between \$0.05 and \$0.10 per HIT, which is competitive with other sense annotation tasks (Akkaya et al., 2010; Hong and Baker, 2011; Rumshisky, 2011; Passonneau and Carpenter, 2014). I finished with 185 annotators producing a total of $14,607$ annotations and 137 instances of feedback.

Some turkers may provide poor annotations, and therefore, an initial pass over the annotations can help identify these turkers (Gella et al., 2014). Gella et al. (2014) included a gold-standard instance within each HIT and disregarded all annotations from turkers that performed poorly on these gold-standard instances. Instead of providing a gold-standard instance within each HIT, I compare each turker's annotations against a majority vote. I avoid the use of a gold standard because not all senses in WordNet have an example. I do this by performing an initial pass over the annotations to calculate the majority vote for each instance and then calculate each annotator's accuracy with respect to the majority

vote. Annotations from any turker that scored less than 50% agreement with the majority vote are removed from consideration, leaving 162 turkers in the dataset. I perform a second pass through the dataset and calculate the majority vote for each instance and remove 7 instances, which were assigned *I cannot assign a sense* as the most frequent label and 2 instances where there was a tie for the most frequent label. The involvement of turkers as annotators was reviewed and approved by the University of New Brunswick's Research Ethics Board and a copy of the ethics application can be seen in Appendix A.1.

5.2.3 Speech Situation

All text was originally obtained from downloading all accessible blogs from `blogger.com` on a single day in August of 2004 (Schler et al., 2006). Table 5.2 shows details about the number of instances across authors and their age. The number of instances ranges from 3 to 152 with a mean of 44 and a median of 31 instances per author. The age of the authors ranges from 17 to 48 years with a mean of 30 and a median of 27. The sex of the authors is disproportionate, with 10 females and 26 males. Information for individual authors can be seen in Appendix A.2.

Metric	# Instances	Age
Min	3	17
Max	152	48
Mean	44	30
Median	31	27

Table 5.2: The details of the dataset with respect to the number of instances across authors in the dataset and the age of the authors.

5.2.4 Speaker Demographic

All text in the original corpus was English, although there was non-English text in the blogs, which was removed by Schler et al. (2006).

5.2.5 Dataset Analysis

The final dataset consists of 11 lemmas and 1586 annotated instances. The instances that are included in the dataset are the same instances that were given to the turkers, which span across sentences. I did this to maintain consistency with the text being annotated by turkers and the text being used by WSD models. Table 5.3 shows the number of instances and assigned senses for each lemma. The number of instances per lemma ranges from 93 for *charge* to 192 for *paper* with an average of 144. The number of assigned senses ranges from 4 for *deal* to 13 for *field* with an average number of 8.5, known as sense ambiguity (Jurgens, 2014). The sense ambiguity is a metric that can be used to measure the difficulty of a WSD dataset. The dataset's sense ambiguity of 8.5 is among the higher values of the datasets in the collection from Raganato et al. (2017) and the dataset from Gella et al. (2014), which ranges from 4.9 to 8.9.

Lemma	# instances	# senses assigned
Paper	192	7
Position	176	12
Sign	163	9
Form	156	10
Case	154	10
Degree	146	6
Track	146	8
Deal	140	4
Field	121	13
Rule	99	6
Charge	93	8
Average	144	8.5

Table 5.3: The number of instances and assigned senses for each lemma.

Figures 5.2 and 5.3 show the sense distributions for each author for two lemmas in the dataset. Depending on the lemma, the authors' sense distributions can be similar to the average sense distribution across all authors (labelled as *All_Authors* in the figures), which is seen in Figure 5.2 for *form*. Other times, such as for *charge* in Figure 5.3, the sense distributions of many authors differ from the average sense distribution across all authors. In Appendix A.3, I show author-level sense distributions for each lemma from the dataset. In Figure 5.4, I show the sense distributions for four authors with respect to the lemma *deal*,

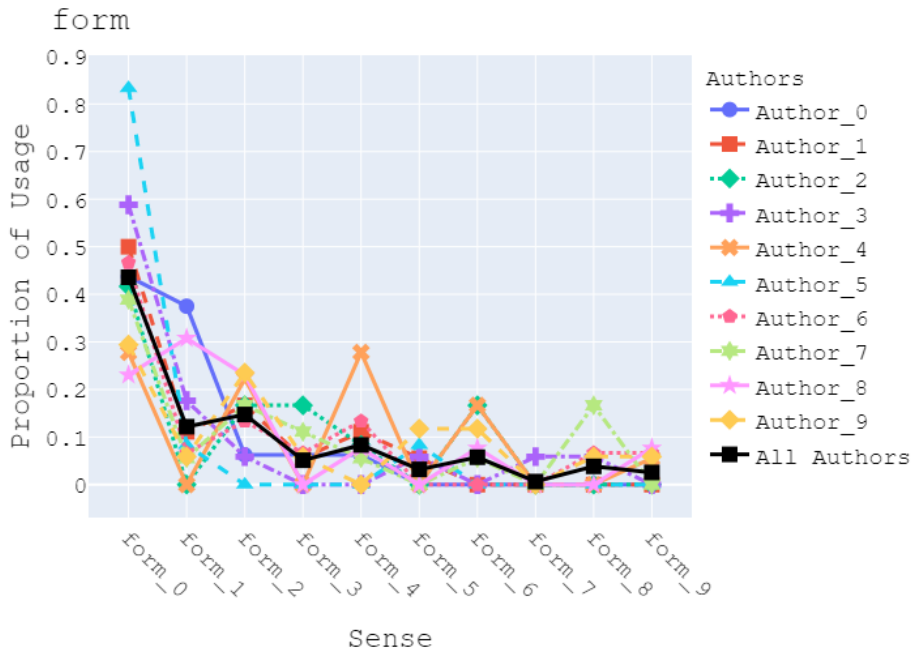


Figure 5.2: Sense distributions for *form*. The average distribution across all authors is in black.

which shows how authors can use the same lemma differently and the potential benefits of tailoring models toward an individual author. Specifically, each author in Figure 5.4 has a different predominant sense for the lemma *deal* and only *Author_0* shares their predominant sense with the predominant sense across all authors.

5.3 Methods

In this section, I discuss the WSD methods that I evaluate on the dataset, which was presented in the previous section. This includes predominant sense baselines, a state-of-the-art method (*SensEmBERT*), and my proposed personalized models.

5.3.1 Baselines

I apply three WSD baselines, which include always predicting the predominant sense for each lemma. The predominant sense is calculated via WordNet first senses (*WORDNET*), the predominant sense of the dataset (*DATASET_PREDOM*), and the predominant sense for each author (*AUTHOR_PREDOM*).

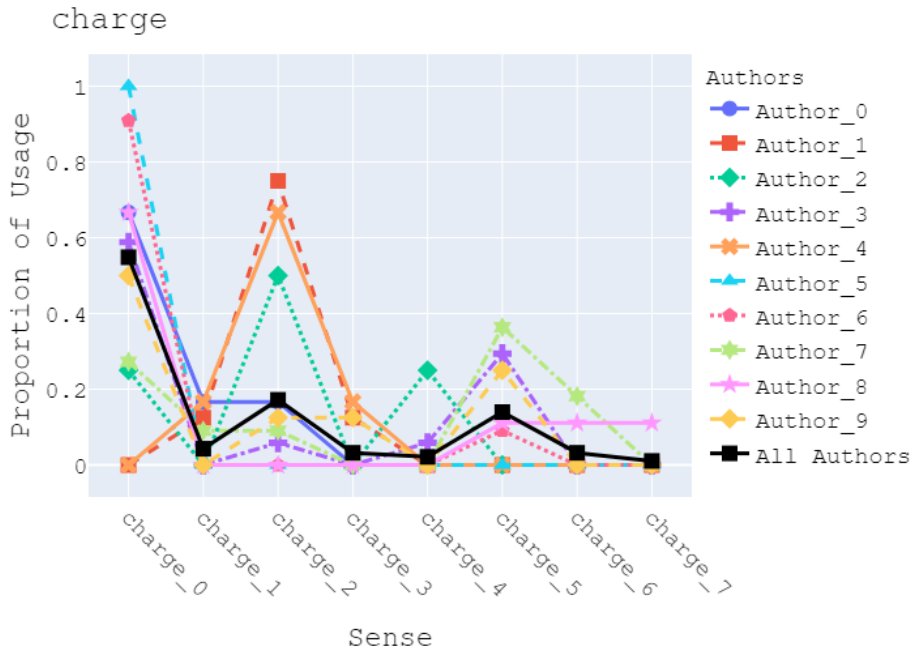


Figure 5.3: Sense distributions for *charge*.

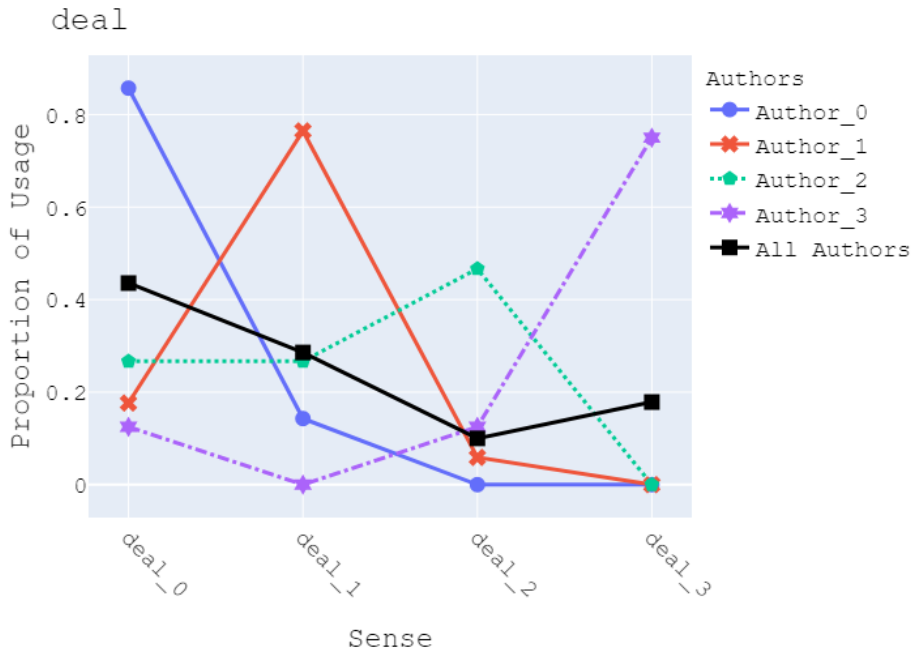


Figure 5.4: Sense distributions for four authors for the lemma *deal*.

5.3.2 SensEmBERT

This method is an unsupervised method that has achieved state-of-the-art results on the English datasets from Raganato et al. (2017) and can outperform supervised WSD models on less frequent lemmas and senses (Scarlini et al., 2020). It uses sense embeddings of a noun by embedding text from Wikipedia articles related to the noun and concatenating it with an embedding of text from the noun's BabelNet entry (Navigli and Ponzetto, 2012). The embeddings are calculated by summing the last four layers of BERT after using a target token in context as input.

SensEmBERT assigns a token's sense by embedding the token with BERT and concatenating this embedding onto itself to double the vector length. Cosine similarity is calculated between this vector and all sense embeddings for all possible senses for the lemma of the target token. The sense that has the highest similarity is assigned as the target token's sense.

5.3.3 Personalizing SensEmBERT

I extend *SensEmBERT* by using text from the author to tailor the model to them by assuming knowledge about the author's sense distributions. These methods are used to explore the benefits of personalized WSD systems and the potential gains of learning the sense distributions of an author. I discuss these types of methods in this subsection.

SEBERT_PERS

In this method, I exploit the Zipfian distribution that sense frequencies tend to exhibit (Kilgarriff, 2004). Specifically, the weights for each sense that are outputted by *SensEmBERT* are ranked and the final score for a sense is calculated by the inverse rank of the sense multiplied by the probability of this sense given an author (calculated by the author's gold standard sense distributions) as seen in Equation 5.1. The sense that results in the highest score is assigned as the sense of the target token.

$$weight(sense) = \frac{1}{rank} * p(sense|author) \tag{5.1}$$

SEBERT_PERS_PREDOM

The author-level sense distribution of a lemma could be difficult to automatically estimate and it might be easier to instead estimate the author-level predominant sense. Therefore, I assume knowledge of only the author-level predominant sense of a lemma for this method by using the author's gold standard predominant sense. Specifically, I assign the sense that was given the most weight according to *SensEmBERT* if the predominant sense of the author is not among the top k ranked senses. If the predominant sense is in the top k ranked senses, I assign the predominant sense. I refer to this k value as the override rank and it is a hyperparameter that needs to be tuned. I also explore the use of the predominant senses from the dataset and WordNet; I refer to these methods as *SEBERT_DATASET_PREDOM* and *SEBERT_WORDNET_PREDOM*, respectively.

5.4 Experimental Results

In this section, I evaluate the models. I first explore the tuning of the override rank for *PREDOM*-based methods. I then evaluate my models using average accuracy across all authors and accuracy for individual authors.

5.4.1 Tuning SEBERT_PERS_PREDOM

Unfortunately, due to the relatively small size of my dataset, I am unable to use a held-out subset of the data for model tuning. Therefore, I show the performance of the *PREDOM*-based models using different override ranks. I select the range 2-6, inclusively, to incorporate the lowest value that would not result in the model constantly predicting *SensEmBERT*'s predicted sense, up to a value that results in *SEBERT_DATASET_PREDOM* performing worse than *SensEmBERT*. Figure 5.5 shows *SEBERT_PERS_PREDOM* and *SEBERT_DATASET_PREDOM* outperform *SensEmBERT* for any of the tested override rank values except for *SEBERT_DATASET_PREDOM* with an override rank of 6. This finding eliminates the need to finetune this model, since any value between 2 and 5, inclusively, works reasonably well. Increasing the override rank results in the model that uses WordNet first-senses to perform worse, which suggests that the authors' predominant senses do not

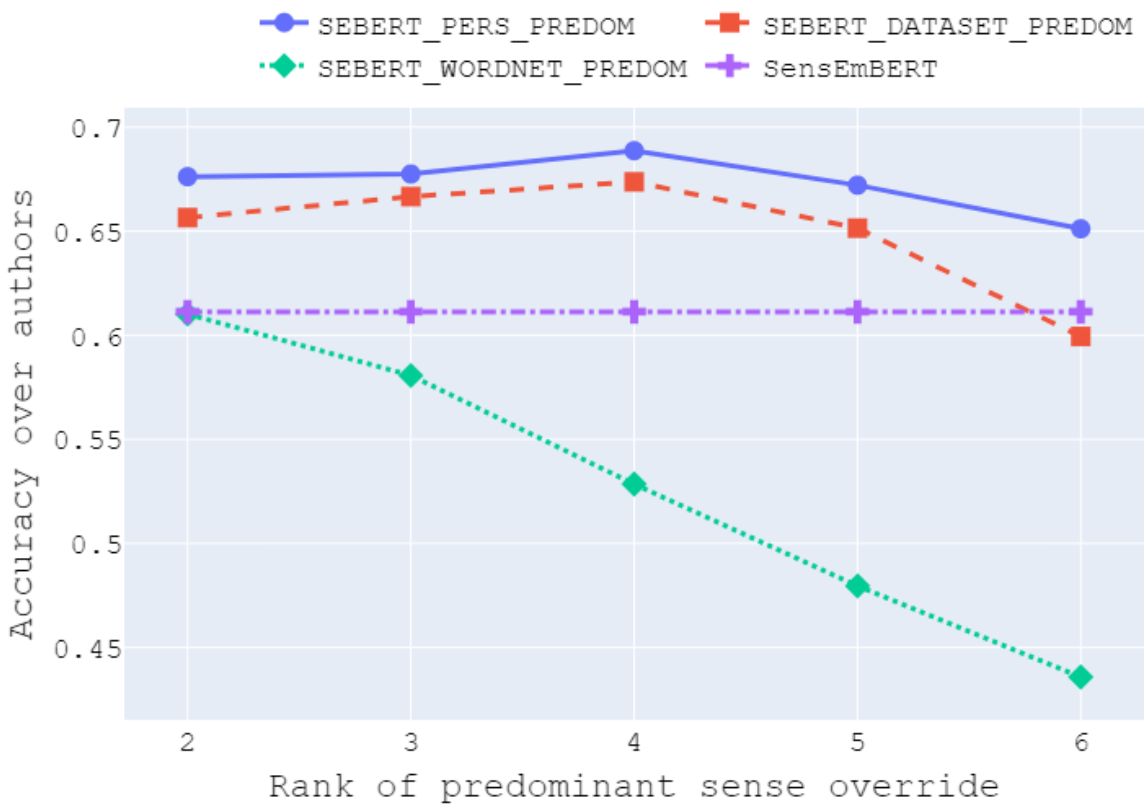


Figure 5.5: Average accuracy across authors for the *PREDOM*-based methods compared to *SensEmBERT*.

align with WordNet's first senses. I do not consider *SEBERT_WORDNET_PREDOM* for the remaining experiments due to its poor performance and I use an override rank value of 4 for *SEBERT_PERS_PREDOM* and *SEBERT_DATASET_PREDOM*.

5.4.2 Overall Accuracy

In this subsection, I discuss the results of each method on the entire dataset. I evaluated each model using accuracy across instances, average accuracy across lemmas, and average accuracy across authors. I evaluated using these three metrics to eliminate the issue of having non-uniform distributions of instances in the dataset. For example, the number of instances per author ranges from 3 to 152 and, therefore, I would like to weight each author equally in the case of accuracy across authors, instead of favouring models that only perform well on authors with more instances in the dataset.

Table 5.4 shows the scores for each method across the different evaluations. The three pre-

Method	Inst	Lem	Auth
WordNet First-Sense	0.204	0.208	0.193
Dataset Predominant sense	0.384	0.395	0.396
Author Predominant sense	0.552	0.560	0.569
SensEmBERT	0.574	0.585	0.611
SEBERT_PERS	0.712	0.716	0.738
SEBERT_PERS_PREDOM	0.656	0.661	0.689
SEBERT_DATASET	0.660	0.655	0.664
SEBERT_DATASET_PREDOM	0.625	0.627	0.674

Table 5.4: Average accuracy across instances (Inst), lemmas (Lem), and authors (Auth).

dominant sense baselines' performances scored in the expected order, such that using the predominant sense of the author outperforms using the predominant sense of the dataset, which outperforms using the first sense from WordNet. *SensEmBERT* outperformed all other baselines. The inclusion of the author's sense distribution in *SEBERT_PERS* and their predominant sense in *SEBERT_PERS_PREDOM* both outperform *SensEmBERT*. *SEBERT_PERS* achieves the highest score of 0.738 in terms of average accuracy across authors, which is an absolute improvement of 0.127 above SensEmBERT. The use of the dataset-level sense distributions in *SEBERT_DATASET* or predominant senses in *SEBERT_DATASET_PREDOM* outperforms *SensEmBERT* but does not outperform *SEBERT_PERS* and *SEBERT_PERS_PREDOM*, which supports the importance of using author-specific data for personalization. Interestingly, *SEBERT_DATASET_PREDOM* outperforms *SEBERT_DATASET* for average accuracy across authors. These findings indicate that SensEmBERT can be improved through personalization by incorporating information about author-level sense distributions or predominant senses.

5.4.3 Author-level Performance

The ability to perform well on each author is a way to evaluate the fairness of a model (Hashimoto et al., 2018; Ethayarajh and Jurafsky, 2020). In this subsection, I consider the models' performance on authors that they perform poorly on, in order to measure their fairness.

Figure 5.6 shows the performance of my two personalized models (*SEBERT_PERS* and *SEBERT_PERS_PREDOM*) and the *SensEmBERT* baseline for each author in the dataset. The authors are sorted in ascending order with respect to the accuracy of *SensEmBERT*. It

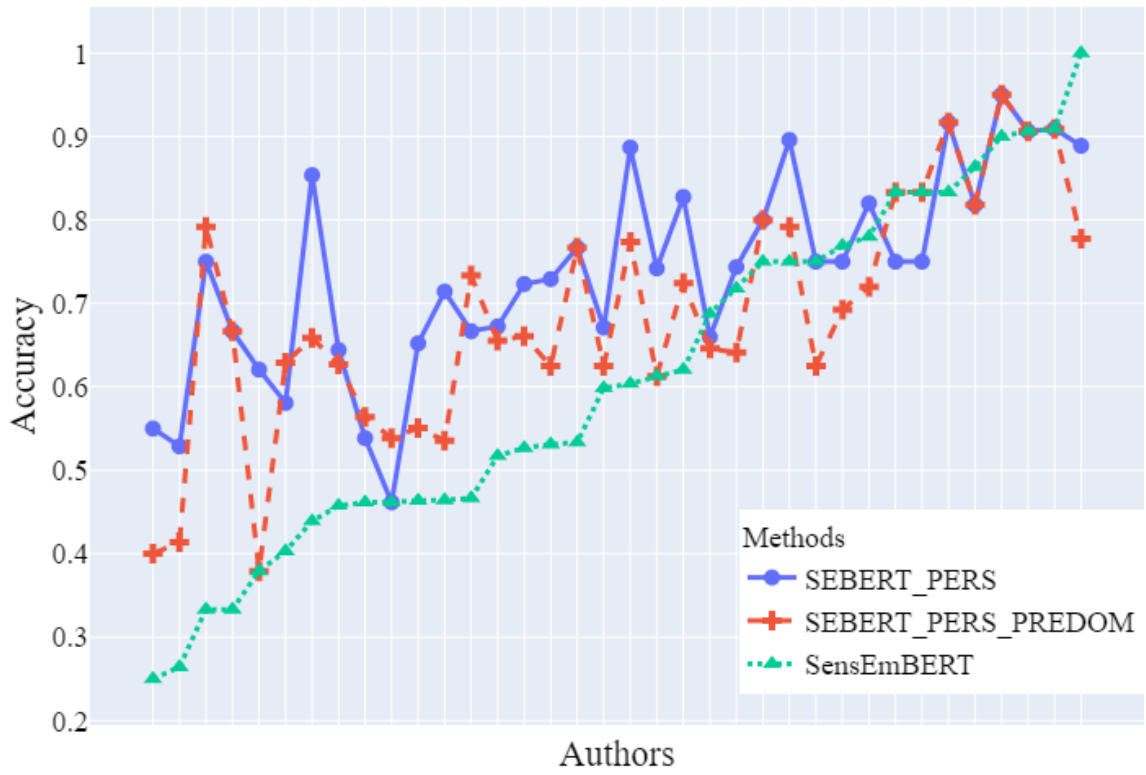


Figure 5.6: Author-level performances for the two personalized methods and *SensEmBERT*.

shows that authors that *SensEmBERT* performs below average on with respect to accuracy across authors (i.e. 0.611) often receive the largest boost in performance from personalized models. This could be due to those individuals having different writing styles as compared with text that *SensEmBERT* was trained on, which is an interesting topic for further exploration. The authors that achieve approximately 0.70 or greater for *SensEmBERT* can have their performance hindered by personalization, although often not by a large amount. *SEBERT_PERS* usually outperforms *SEBERT_PERS_PREDOM* for a given author.

One of the main pillars of my work is to provide personalized models that work well for authors that scored poorly with conventional non-personalized models. Therefore, I would ideally want models that do not perform poorly on any single author, which I can evaluate by averaging the accuracy over the authors that each model achieves the lowest accuracy on. In Figure 5.7, I evaluate my models on the x authors that each model achieves their lowest accuracy on for values of x ranging from 1 author to all 36 authors.

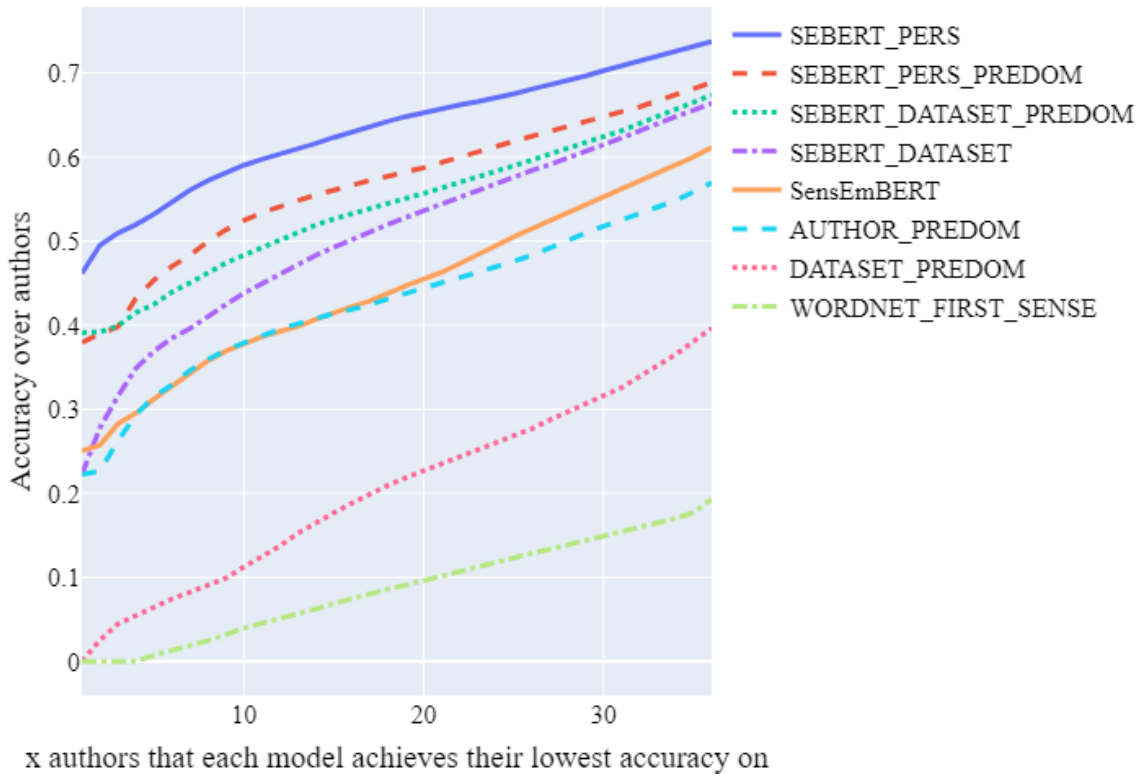


Figure 5.7: Performance of methods on the x authors that they achieve the lowest accuracy on.

It shows that *SensEmBERT* achieves 0.25 for the author on which it performs worst, while *SEBERT_PERS* achieves 0.46 for the author on which it performs worst. Furthermore, *SEBERT_PERS*, *SEBERT_PERS_PREDOM*, and *SEBERT_DATASET_PREDOM* always outperform *SensEmBERT* for every value of x when evaluating on the x authors that each model achieves their lowest accuracy on, with *SEBERT_PERS* always achieving the highest score. This finding demonstrates that personalized WSD models such as *SEBERT_PERS* and *SEBERT_PERS_PREDOM* are more fair than non-personalized models (*SensEmBERT*).

5.5 Summary

In this chapter, I proposed a novel dataset for personalized WSD and showed that sense distributions and predominant senses of an author can be used to personalize an existing knowledge-based WSD model (*SensEmBERT*). My experiments consistently show that

models that consider author-specific sense distributions (*SEBERT_PERS*) or predominant senses (*SEBERT_PERS_PREDOM*) can outperform models that do not consider any knowledge of sense distributions (*SensEmBERT*). Furthermore, I show that models that use author-level sense distributions or predominant senses outperform models that use genre-level sense distributions (*SEBERT_DATASET*) or predominant senses (*SEBERT_DATASET_PREDOM*). *SEBERT_PERS* achieved the highest accuracy across all authors with an absolute improvement of 0.127 over *SensEmBERT*. I further explore the fairness of my models by evaluating their accuracy on authors that they perform poorly on and showed that the lowest accuracy achieved by *SEBERT_PERS* on a single author is 0.21 above *SensEmBERT's* lowest accuracy, which indicates that personalization can produce a more fair WSD system. My work shows the importance of learning sense distributions of individual authors for WSD, and therefore, should be considered when developing WSD systems.

Chapter 6

Conclusions

In this chapter, I conclude my thesis and suggest further avenues to explore for each studied topic. People tend to develop their own use of a language, which is unique for each individual. This phenomenon was seen in Chapter 5 by exploring the difference in predominant senses of different authors. In many NLP tasks, models often do not train on text from a single individual, but they usually train on text that is constructed by combining the text from multiple authors such as the background language models used in Chapter 3, the word embeddings used in Chapter 4, and the domain-based word sense disambiguation methods used in Chapter 5. Throughout my thesis, I have tested my hypothesis that personalized models will outperform models that are not personalized.

In concluding my work, I revisit the three research questions that this thesis attempts to answer.

1. **Do personalized language models outperform models that are not tailored toward an individual?** In Chapter 3, I showed that priming an LSTM language model on a relatively small amount of text from a single individual can outperform a non-personalized model when estimating probabilities over a sequence of words. This statement remains true for three different evaluation metrics, which include adjusted perplexity, accuracy @ k, and accuracy @ k given c. In this chapter, I also showed that if enough text is available, $100k+$ tokens, the interpolation of a background model with a model trained on only text from the author can outperform a model that uses the priming technique.

2. **Can personalized language models outperform previously state-of-the-art**

embedding-based models, on the task of authorship verification? Following the success of personalizing language models, I recruited personalized language models for an authorship verification task in Chapter 4. I showed that a personalized language model, which used the priming technique, greatly outperforms the previously state-of-the-art embedding-based model, while only requiring a minuscule amount of text from the author. Even when the embedding-based model had access to $100k$ training tokens from an author, the personalized language model achieved better results with only 100 tokens.

3. **Can personalized word sense disambiguation models outperform models that are not tailored toward a single author?** I started my discussion in Chapter 5 with a detailed description of my proposed personalized word sense disambiguation dataset. This included a description of how the dataset was created and an analysis of the data. This dataset will enable further research on personalized word sense disambiguation. I extended a state-of-the-art knowledge-based method (SensEmBERT) by incorporating author-level information in the form of their sense distributions or their predominant senses. By including this author-level information in a personalized model, I was able to outperform SenseEmBERT. I further explored the use of sense distributions from a collection of authors and found that the personalized model still achieved better results. Furthermore, the personalized models exhibited more fairness, which was evaluated by considering the authors that each model performed poorly on.

For each topic studied in this thesis (language modeling, authorship verification, and word sense disambiguation), I showed how personalization can benefit an individual author regardless of their demographic and while requiring relatively low amounts of text from them. This can potentially assist any individual with downstream applications that depend on models from these three topics.

6.1 Future Work

In this section, I suggest immediate next steps for each of the topics that I discussed and then I focus on future work for much larger topics.

For personalized language modeling, I considered LSTM and n-gram language models, but

there is a newer language model (BERT) (Devlin et al., 2019) that might perform well when personalized. However, the priming technique that was used to prime the LSTM-based language models will not be applicable for BERT due to the difference in structures. Specifically, LSTMs maintain some level of information of text that it has seen within its state, which is what the priming method exploits, but BERT uses a fixed input of text and does not recall information of previously seen text. However, an author-level matrix could be incorporated into BERT's framework, similar to how Jaech and Ostendorf (2017) adapt the weights of a recurrent neural network-based language model with author-level matrices to produce a personalized BERT model.

For authorship verification, I focused on detecting the difference between text from a given author and text that contains one of three different types of malicious content, which includes email spam, hate speech, and erotica. This work could be extended to include different types of malicious content such as incriminating text (text that could prompt legal actions such as death threats or slander) or text that goes against the author's beliefs, such as their stance on controversial issues.

For personalized word sense disambiguation, I proposed models that utilize either the sense distributions of an author or their predominant senses and found that both were able to greatly outperform a state-of-the-art model that does not consider author-level information. This finding demonstrates the importance of being able to estimate the sense distributions of an author or the predominant senses of an author. I leave the task of finding the sense distributions of an author as the suggested future work for personalized word sense disambiguation. Some of the more recently applied methods for learning sense distributions include a cluster-based approach that uses BERT (Pasini et al., 2020) and a topic modeling approach (Bennett et al., 2016). Such approaches could potentially be applied to learn author-level sense distributions and predominant senses.

The success of personalization in language modeling, authorship verification, and word sense disambiguation suggests that personalization could benefit other NLP tasks, such as sentiment analysis (Turney, 2002), readability (Dale and Chall, 1949), and humour detection and rating (Meaney et al., 2021). Sentiment analysis involves determining the sentiment in text, such as a review on a product. I showed that the predominantly used

sense for a lemma can differ among authors, therefore, the sentiment that is predominantly expressed for each lemma could also differ, and therefore, sentiment analysis could potentially benefit from personalization. For example, the phrase *That was sick* could be expressing a positive sentiment or a negative sentiment. Similar to how I used word sense distributions to personalize word sense disambiguation, the sentiment distributions for an author could potentially assist a sentiment analysis model for a single individual.

Readability involves rating the reading difficulty of a piece of text. This is especially useful for measuring the difficulty of text for people learning a second language (Xia et al., 2016). The readability of text can depend on the reader. For example, a document that discusses artificial neural networks could be fairly simple to understand for someone who studies machine learning, but might be fairly difficult for someone studying a non-computer science related field. Personalization could be used to evaluate the readability of a document, while considering a specific individual. The vocabulary of an author, i.e., the words that they have previously used, could be used to assist a readability model by rating words that are not similar to words in the author's vocabulary as being more difficult.

Lastly, humour detection and rating is a fairly new topic, which involves predicting if a snippet expresses humour and rating the level of humour if it does. It is likely that almost everyone has met a person that differed on what they consider humorous. For example, some people might enjoy jokes that revolve around pop culture more than jokes that are relatable to programmers, and therefore, the humour rating of a snippet is dependent on the reader's sense-of-humour. A personalized model can consider what a specific individual considers to exhibit humour while rating text. Specifically, if a model was able to learn what topics an author writes about then a model could potentially use this to assist in classifying a snippet of text as being humorous or not.

To summarize, I believe that personalization in these three topics should be studied and through personalization, there is the possibility to create better language technology for all people.

Bibliography

Oliver Adams, Adam Makarucha, Graham Neubig, Steven Bird, and Trevor Cohn. 2017. Cross-lingual word embeddings for low-resource language modeling. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*. Association for Computational Linguistics, Valencia, Spain, pages 937–947.

Cem Akkaya, Alexander Conrad, Janyce Wiebe, and Rada Mihalcea. 2010. Amazon mechanical turk for subjectivity word sense disambiguation. In *Proceedings of the NAACL HLT 2010 workshop on creating speech and language data with Amazon's Mechanical Turk*. Los Angeles, USA, pages 195–203.

Firoj Alam, Shafiq Joty, and Muhammad Imran. 2018. Domain adaptation with adversarial training and graph embeddings. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Melbourne, Australia, pages 1077–1087.

Silvio Amir, Byron C. Wallace, Hao Lyu, Paula Carvalho, and Mario J. Silva. 2016. Modelling context with user embeddings for sarcasm detection in social media. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*. Association for Computational Linguistics, Berlin, Germany, pages 167–177.

Satanjeev Banerjee and Ted Pedersen. 2002. An adapted Lesk algorithm for word sense disambiguation using WordNet. In *Proceedings of the 3rd International Conference on Intelligent Text Processing and Computational Linguistics (CICLing-2002)*. Mexico City, Mexico, pages 136–145.

Pierpaolo Basile, Annalina Caputo, and Giovanni Semeraro. 2014. An enhanced lesk word sense disambiguation algorithm through a distributional semantic model. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*. Dublin City University and Association for Computational Linguistics, Dublin, Ireland, pages 1591–1600.

Emily M. Bender and Batya Friedman. 2018. Data statements for natural language processing: Toward mitigating system bias and enabling better science. *Transactions of the Association for Computational Linguistics* 6:587–604.

José-Miguel Benedi and Joan-Andreu Sánchez. 2000. Combination of n-grams and stochastic context-free grammars for language modeling. In *Proceedings of the 18th conference on Computational linguistics-Volume 1*. Association for Computational Linguistics, Centre Universitaire, Luxembourg, pages 55–61.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of machine learning research* 3(Feb):1137–1155.

Andrew Bennett, Timothy Baldwin, Jey Han Lau, Diana McCarthy, and Francis Bond. 2016. LexSemTm: A semantic dataset based on all-words unsupervised sense distribution learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 1513–1524.

Michele Bevilacqua and Roberto Navigli. 2020. Breaking through the 80in word sense disambiguation by incorporating knowledge graph information. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Online, pages 2854–2864.

Steven Bird, Edward Loper, and Ewan Klein. 2009. *Natural Language Processing with Python*. O'Reilly Media Inc.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent dirichlet allocation. *Journal of Machine Learning Research* 3:993–1022.

Jordan L. Boyd-Graber, David M. Blei, and Xiaojin Zhu. 2007. A topic model for word sense disambiguation. In *EMNLP-CoNLL*. Prague, Czech Republic.

Ciprian Chelba and Frederick Jelinek. 2000. Structured language modeling. *Comput. Speech Lang.* 14(4):283–332.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. 2013. One billion word benchmark for measuring progress in statistical language modeling. *arXiv preprint arXiv:1312.3005* .

Timothy Chklovski and Rada Mihalcea. 2002. Building a sense tagged corpus with open mind word expert. In *Proceedings of the ACL-02 workshop on Word sense disambiguation: recent successes and future directions*. Philadelphia, USA, pages 116–122.

Do Kook Choe and Eugene Charniak. 2016. Parsing as language modeling. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Austin, Texas, pages 2331–2336.

Edgar Dale and Jeanne S Chall. 1949. The concept of readability. *Elementary English* 26(1):19–26.

Ona de Gibert, Naiara Perez, Aitor García-Pablos, and Montse Cuadros. 2018. Hate speech dataset from a white supremacy forum. In *Proceedings of the 2nd Workshop on Abusive Language Online (ALW2)*. Brussels, Belgium, pages 11–20.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Association for Computational Linguistics, Minneapolis, Minnesota, pages 4171–4186.

Javid Ebrahimi and Dejing Dou. 2016. Personalized semantic word vectors. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*. ACM, Indianapolis, Indiana, USA, CIKM '16, pages 1925–1928.

Philip Edmonds and Scott Cotton. 2001. SENSEVAL-2: Overview. In *Proceedings of SENSEVAL-2 Second International Workshop on Evaluating Word Sense Disambiguation Systems*. Association for Computational Linguistics, Toulouse, France, pages 1–5.

Jesse Egbert, Douglas Biber, and Mark Davies. 2015. Developing a bottom-up, user-based method of web register classification. *Journal of the Association for Information Science and Technology* 66(9):1817–1831.

Katrin Erk, Diana McCarthy, and Nicholas Gaylord. 2009. Investigations on word senses and word usages. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*. Association for Computational Linguistics, Suntec, Singapore, pages 10–18.

Kawin Ethayarajh and Dan Jurafsky. 2020. Utility is in the eye of the user: A critique of nlp leaderboard design. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Online, pages 4846–4853.

Lucie Flek. 2020. Returning the N to NLP: Towards contextually personalized classification models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Online, pages 7828–7838.

William A. Gale, Kenneth W. Church, and David Yarowsky. 1992. One sense per discourse. In *Proceedings of the Workshop on Speech and Natural Language*. Association for Computational Linguistics, Harriman, New York, HLT '91, pages 233–237.

Spandana Gella, Paul Cook, and Timothy Baldwin. 2014. One sense per tweeter ... and other lexical semantic tales of Twitter. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics, volume 2: Short Papers*. Gothenburg, Sweden, pages 215–220.

Tatsunori Hashimoto, Megha Srivastava, Hongseok Namkoong, and Percy Liang. 2018. Fairness without demographics in repeated loss minimization. In *International Conference on Machine Learning*. PMLR, Stockholm, Sweden, pages 1929–1938.

Kenneth Heafield, Ivan Pouzyrevsky, Jonathan H. Clark, and Philipp Koehn. 2013. Scalable modified Kneser-Ney language model estimation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*. Sofia, Bulgaria, pages 690–696.

Jisup Hong and Collin F Baker. 2011. How good is the crowd at "real" wsd? In *Proceedings of the 5th linguistic annotation workshop*. Portland, USA, pages 30–37.

Dirk Hovy. 2015. Demographic factors improve classification performance. In *Proceedings of the 53rd annual meeting of the Association for Computational Linguistics and the 7th international joint conference on natural language processing (volume 1: Long papers)*. Beijing, China, pages 752–762.

Eduard Hovy, Mitch Marcus, Martha Palmer, Lance Ramshaw, and Ralph Weischedel. 2006. Ontonotes: the 90% solution. In *Proceedings of the human language technology conference of the NAACL, Companion Volume: Short Papers*. New York City, USA, pages 57–60.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Melbourne, Australia, pages 328–339.

Eric Huang, Richard Socher, Christopher Manning, and Andrew Ng. 2012. Improving word representations via global context and multiple word prototypes. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Jeju Island, Korea, pages 873–882.

Xiaolei Huang and Michael J. Paul. 2019. Neural user factor adaptation for text classification: Learning to generalize across author demographics. In *Proceedings of the Eighth Joint Conference on Lexical and Computational Semantics (*SEM 2019)*. Association for Computational Linguistics, Minneapolis, Minnesota, pages 136–146.

Kazuki Irie, Shankar Kumar, Michael Nirschl, and Hank Liao. 2018. Radmm: Recurrent

adaptive mixture model with applications to domain robust language modeling. *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* pages 6079–6083.

Aaron Jaech and Mari Ostendorf. 2017. Low-rank rnn adaptation for context-aware language modeling. *Transactions of the Association for Computational Linguistics* 6.

Aaron Jaech and Mari Ostendorf. 2018. Personalized language model for query autocompletion. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Melbourne, Australia, pages 700–705.

Magdalena Jankowska, Evangelos Milios, and Vlado Keselj. 2014. Author verification using common n-gram profiles of text documents. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*. Dublin City University and Association for Computational Linguistics, Dublin, Ireland, pages 387–397.

S. Ji, S. Pan, G. Long, X. Li, J. Jiang, and Z. Huang. 2019. Learning private neural language modeling with attentive aggregation. In *2019 International Joint Conference on Neural Networks (IJCNN)*. pages 1–8.

David Jurgens. 2014. An analysis of ambiguity in word sense annotations. In *Proceedings of Ninth International Conference on Language Resources and Evaluation*. Citeseer, Reykjavik, Iceland, pages 3006–3012.

Urvashi Khandelwal, He He, Peng Qi, and Dan Jurafsky. 2018. Sharp nearby, fuzzy far away: How neural language models use context. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Melbourne, Australia, pages 284–294.

Adam Kilgarriff. 2004. How dominant is the commonest sense of a word? In *International conference on text, speech and dialogue*. Springer, Brno, Czech Republic, pages 103–111.

Adam Kilgarriff and Joseph Rosenzweig. 2000. English senseval: Report and results. In

Proceedings of the Second International Conference on Language Resources and Evaluation (LREC'00). European Language Resources Association (ELRA), Athens, Greece.

Milton King, Dima Alhadidi, and Paul Cook. 2018. Text-based detection of unauthorized users of social media accounts. In Ebrahim Bagheri and Jackie C.K. Cheung, editors, *Advances in Artificial Intelligence. Canadian AI 2018. Lecture Notes in Computer Science.*. Springer, Toronto, Canada, volume 10832, pages 292–297.

Milton King and Paul Cook. 2019. Building personalized language models through language model interpolation. In *Proceedings of the 20th International Conference on Computational Linguistics and Intelligent Text Processing (CICLing 2019)*. La Rochelle, France.

Milton King and Paul Cook. 2020a. Authorship verification with personalized language models. In *International Conference on Text, Speech, and Dialogue*. Springer, Online, pages 248–256.

Milton King and Paul Cook. 2020b. Evaluating approaches to personalizing language models. In *Proceedings of the 12th Language Resources and Evaluation Conference*. European Language Resources Association, Marseille, France, pages 2461–2469.

Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S. Zemel, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. 2015. Skip-thought vectors. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, Curran Associates, Inc., pages 3276–3284.

Rob Koeling, Diana McCarthy, and John Carroll. 2005. Domain-specific sense distributions and predominant sense acquisition. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP 2005)*. Vancouver, Canada, pages 419–426.

Veronika Laippala, Samuel Rönnqvist, Saara Hellström, Juhani Luotolahti, Liina Repo, Anna Salmela, Valtteri Skantsi, and Sampo Pyysalo. 2020. From web crawl to clean register-annotated corpora. In *Proceedings of the 12th Web as Corpus Workshop*. European Language Resources Association, Marseille, France, pages 14–22.

Jey Han Lau, Paul Cook, Diana McCarthy, Spandana Gella, and Timothy Baldwin. 2014. Learning word sense distributions, detecting unattested senses and identifying novel senses using topic models. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL 2014)*. Baltimore, USA, pages 259–270.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on Machine Learning*. Beijing, China.

Jiwei Li, Michel Galley, Chris Brockett, Georgios Spithourakis, Jianfeng Gao, and Bill Dolan. 2016. A persona-based neural conversation model. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 994–1003.

Linlin Li, Benjamin Roth, and Caroline Sporleder. 2010. Topic models for word sense disambiguation and token-based idiom detection. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*. Uppsala, Sweden, pages 1138–1147.

Thang Luong, Richard Socher, and Christopher Manning. 2013. Better word representations with recursive neural networks for morphology. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*. Association for Computational Linguistics, Sofia, Bulgaria, pages 104–113.

Kim Luyckx and Walter Daelemans. 2008. Authorship attribution and verification with many authors and limited data. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*. Coling 2008 Organizing Committee, Manchester, UK, pages 513–520.

Veronica Lynn, Youngseo Son, Vivek Kulkarni, Niranjan Balasubramanian, and H. Andrew Schwartz. 2017. Human centered NLP with user-factor adaptation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Copenhagen, Denmark, pages 1157–1166.

Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The Stanford CoreNLP Natural Language Processing Toolkit. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*. Baltimore, USA, pages 55–60.

M. Marcus, G. Kim, M. Marcinkiewicz, R. MacIntyre, A. Bies, M. Ferguson, K. Katz, and B. Schasberger. 1994. The Penn Treebank: Annotating predicate argument structure. In *Proceedings of the ARPA Human Language Technology Workshop*. Plainsboro, USA.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics* 19(2):313–330.

U-V Marti and Horst Bunke. 2001. On the influence of vocabulary size and language models in unconstrained handwritten text recognition. In *Proceedings of Sixth International Conference on Document Analysis and Recognition*. IEEE, Seattle, USA, pages 260–265.

Diana McCarthy, Rob Koeling, Julie Weeds, and John Carroll. 2004. Finding predominant word senses in untagged text. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04), Main Volume*. Barcelona, Spain, pages 279–286.

Diana McCarthy, Rob Koeling, Julie Weeds, and John Carroll. 2007. Unsupervised acquisition of predominant word senses. *Computational Linguistics* 33(4):553–590.

H. Brendan McMahan, Daniel Ramage, Kunal Talwar, and Li Zhang. 2017. Learning differentially private language models without losing accuracy. *CoRR* abs/1710.06963.

J.A. Meaney, Steve Wilson, Luis Chiruzzo, Adam Lopez, and Walid Magdy. 2021. Semeval 2021 task 7: Hahackathon, detecting and rating humor and offense. In *Proceedings of the 15th International Workshop on Semantic Evaluation (SemEval 2021)*.

Oren Melamud, Jacob Goldberger, and Ido Dagan. 2016. context2vec: Learning generic context embedding with bidirectional lstm. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*. Berlin, Germany, pages 51–61.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models. *CoRR* abs/1609.07843.

Vangelis Metsis, Ion Androutsopoulos, and Georgios Paliouras. 2006. Spam filtering with naive Bayes-which naive Bayes? In *Proceedings of The Third Conference on Email and Anti-Spam 2006*. California, USA, volume 17, pages 28–69.

Paul Michel and Graham Neubig. 2018. Extreme adaptation for personalized neural machine translation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, Melbourne, Australia, pages 312–318.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *Proceedings of Workshop at the International Conference on Learning Representations, 2013*. Scottsdale, USA.

Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *INTERSPEECH 2010*. Makuhari, Japan, pages 1045–1048.

Tomas Mikolov and Geoffrey Zweig. 2012. Context dependent recurrent neural network language model. In *2012 IEEE Spoken Language Technology Workshop (SLT)*. IEEE, Miami, USA, pages 234–239.

George A. Miller. 1995. Wordnet: A lexical database for english. *Commun. ACM* 38(11):39–41.

George A. Miller, Martin Chodorow, Shari Landes, Claudia Leacock, and Robert G. Thomas. 1994. Using a semantic concordance for sense identification. In *Human Language Technology: Proceedings of a Workshop held at Plainsboro, New Jersey, March 8-11, 1994*. New Jersey, USA.

Andrea Moro and Roberto Navigli. 2015. SemEval-2015 task 13: Multilingual all-words sense disambiguation and entity linking. In *Proceedings of the 9th International Work-*

shop on Semantic Evaluation (SemEval 2015). Association for Computational Linguistics, Denver, Colorado, pages 288–297.

Roberto Navigli, David Jurgens, and Daniele Vannella. 2013. SemEval-2013 task 12: Multilingual word sense disambiguation. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*. Association for Computational Linguistics, Atlanta, Georgia, USA, pages 222–231.

Roberto Navigli and Simone Paolo Ponzetto. 2012. Babelnet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. *Artificial intelligence* 193:217–250.

Franz Josef Och and Hermann Ney. 2004. The alignment template approach to statistical machine translation. *Computational Linguistics* 30(4):417–449.

Tommaso Pasini, Federico Scozzafava, and Bianca Scarlini. 2020. CluBERT: A cluster-based approach for learning sense distributions in multiple languages. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Online, pages 4008–4018.

Rebecca J. Passonneau and Bob Carpenter. 2014. The benefits of a model of annotation. *Transactions of the Association for Computational Linguistics* 2:311–326.

Douglas B. Paul and Janet M. Baker. 1992. The design for the wall street journal-based csr corpus. In *Proceedings of the Workshop on Speech and Natural Language*. Association for Computational Linguistics, Harriman, New York, HLT '91, pages 357–362.

Thomas Plötz and Gernot A. Fink. 2009. Markov models for offline handwriting recognition: a survey. *International Journal on Document Analysis and Recognition (IJDAR)* 12(4):269.

Sameer Pradhan, Edward Loper, Dmitriy Dligach, and Martha Palmer. 2007. SemEval-2007 task-17: English lexical sample, SRL and all words. In *Proceedings of the Fourth*

International Workshop on Semantic Evaluations (SemEval-2007). Association for Computational Linguistics, Prague, Czech Republic, pages 87–92.

Peng Qi, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D. Manning. 2020. Stanza: A Python natural language processing toolkit for many human languages. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*. Online.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners .

Alessandro Raganato, Jose Camacho-Collados, and Roberto Navigli. 2017. Word sense disambiguation: A unified evaluation framework and empirical comparison. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*. Association for Computational Linguistics, Valencia, Spain, pages 99–110.

Adwait Ratnaparkhi. 1996. A maximum entropy model for part-of-speech tagging. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Philadelphia, PA, pages 133–142.

Anna Rumshisky. 2011. Crowdsourcing word sense definition. In *Proceedings of the 5th linguistic annotation workshop*. Portland, USA, pages 74–81.

Alexandre Salle and Aline Villavicencio. 2018. Restricted recurrent neural tensor networks: Exploiting word frequency and compositionality. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, Melbourne, Australia, pages 8–13.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. In *NeurIPS EMC^2 Workshop*. Vancouver, Canada.

Bianca Scarlini, Tommaso Pasini, and Roberto Navigli. 2020. Sensembert: Context-

enhanced sense embeddings for multilingual word sense disambiguation. *Proceedings of the AAAI Conference on Artificial Intelligence* 34(05):8758–8765.

Jonathan Schler, Moshe Koppel, Shlomo Argamon, and James W Pennebaker. 2006. Effects of age and gender on blogging. In *AAAI spring symposium: Computational approaches to analyzing weblogs*. Palo Alto, USA, volume 6, pages 199–205.

Michael R. Schmid, Farkhund Iqbal, and Benjamin C.M. Fung. 2015. E-mail authorship attribution using customized associative classification. *Digit. Investig.* 14(S1):S116–S126.

Benjamin Snyder and Martha Palmer. 2004. The English all-words task. In *Proceedings of SENSEVAL-3, the Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text*. Association for Computational Linguistics, Barcelona, Spain, pages 41–43.

Efstathios Stamatatos, Walter Daelemans, Ben Verhoeven, Patrick Juola, Aurelio López-López, Martin Potthast, and Benno Stein. 2015. Overview of the author identification task at pan 2015. In *CLEF*. Toulouse, France.

Efstathios Stamatatos, Walter Daelemans, Ben Verhoeven, Martin Potthast, Benno Stein, Patrick Juola, Miguel A. Sanchez-perez, and Alberto Barrón-cedeño. 2013. E.: Overview of the author identification task at pan-2013. In *Notebook Papers of CLEF 2013 LABs and Workshops (CLEF-2013) (2013*. Valencia, Spain.

Martin Sundermeyer, Ralf Schlüter, and Hermann Ney. 2012. LSTM neural networks for language modeling. In *INTERSPEECH 2012*. Portland, USA, pages 194–197.

Kaveh Taghipour and Hwee Tou Ng. 2015. One million sense-tagged instances for word sense disambiguation and induction. In *Proceedings of the Nineteenth Conference on Computational Natural Language Learning*. Association for Computational Linguistics, Beijing, China, pages 338–344.

Jaime Teevan, Susan T. Dumais, and Eric Horvitz. 2005. Personalizing search via automated analysis of interests and activities. In *Proceedings of the 28th Annual Interna-*

tional ACM SIGIR Conference on Research and Development in Information Retrieval. Association for Computing Machinery, New York, NY, USA, SIGIR '05, page 449–456.

Peter Turney. 2002. Thumbs up or thumbs down? Semantic orientation applied to unsupervised classification of reviews. *Computing Research Repository - CORR* pages 417–424.

Joerg P. Ueberla. 1994. Analyzing and improving statistical language models for speech recognition. *CoRR* abs/cmp-lg/9406027.

Hans van Halteren. 2004. Linguistic profiling for author recognition and verification. In *Proceedings of the 42Nd Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics, Barcelona, Spain, ACL '04.

Svitlana Volkova, Theresa Wilson, and David Yarowsky. 2013. Exploring demographic language variations to improve multilingual sentiment analysis in social media. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Seattle, USA, pages 1815–1827.

Charles Welch, Jonathan K. Kummerfeld, Verónica Pérez-Rosas, and Rada Mihalcea. 2020. Compositional demographic word embeddings. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Online, pages 4076–4089.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, R'emi Louf, Morgan Funtowicz, and Jamie Brew. 2019. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv* abs/1910.03771.

Menglin Xia, Ekaterina Kochmar, and Ted Briscoe. 2016. Text readability assessment for second language learners. In *Proceedings of the 11th Workshop on Innovative Use of NLP for Building Educational Applications*. Association for Computational Linguistics, San Diego, CA, pages 12–22.

Seunghyun Yoon, H. Yun, Yuna Kim, G. Park, and K. Jung. 2017. Efficient transfer

learning schemes for personalized language modeling using recurrent neural network. *ArXiv* abs/1701.03578.

Steven Young. 1996. A review of large-vocabulary continuous-speech. *IEEE Signal Processing Magazine* 13(5):45–.

Jian Zhang, Xiaofeng Wu, Andy Way, and Qun Liu. 2016. Fast gated neural domain adaptation: Language model as a case study. In *COLING*. Osaka, Japan.

Sen Zhang and Na Dong. 2003. An effective combination of different order n-grams. In *Proceedings of the 17th Pacific Asia Conference on Language, Information and Computation*. Sentosa, Singapore, pages 251–256.

Will Y. Zou, Richard Socher, Daniel Cer, and Christopher D. Manning. 2013. Bilingual word embeddings for phrase-based machine translation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Seattle, USA, pages 1393–1398.

Geoffrey Zweig and Chris J. C. Burges. 2012. A challenge set for advancing language modeling. In *Proceedings of the NAACL-HLT 2012 Workshop: Will We Ever Really Replace the N-gram Model? On the Future of Language Modeling for HLT*. Association for Computational Linguistics, Montreal, Canada, WLM '12, pages 29–36.

Appendix A

Appendix A - Personalized Word Sense Disambiguation Dataset

A.1 REB Form

INSTRUCTIONS FOR APPLYING FOR REVIEW OF RESEARCH INVOLVING HUMANS

REVISED August 2017

DO NOT ATTACH THIS SHEET TO YOUR APPLICATION

Application: The application form that follows is an MS Word file that may be edited by the applicant to provide additional space for answers. **It is helpful if material inserted by the applicant is in bold type. Handwritten applications are not acceptable. On the Fredericton campus, completed applications and attachments should be submitted to the Renée Audet-Martel, REB Coordinator, Office of the Vice President (Research); on the Saint John campus, submissions should be sent to Judith Arseneau, Secretary to the UNBSJ REB, c/o Office of Research Services, Hazen Hall, Room 321A.**

Number of copies required:

Two (2) copies of the application form (**1 with original signatures** and 1 copy)
One (1) copy of an application for external funding (where applicable);
One (1) copy of a thesis proposal (where applicable); and
Two (2) copies of all other attachments.

Applications that do not meet this requirement will be returned as incomplete.

Required attachments: Consult the checklist for a complete list of documents that may be required to be attached to your application.

Electronic Text of Summary: The Research Ethics Board requires submission, by e-mail, of a copy of the Summary of the proposed research (Item 1 on the application form). On the Fredericton campus, this e-mail should be addressed to the Renée Audet-Martel, REB Coordinator, at ethics@unb.ca; on the Saint John campus; it should be addressed to Judith Arseneau, Secretary of the UNBSJ REB reb@unb.ca.

Required Signatures: Please ensure that all required signatures are provided and original. **In particular, note the requirement for approval by your academic unit.**

Incomplete applications: Incomplete applications will be returned to the applicant without review.

Assistance: For assistance in submitting an application for review consult the Chair of the REB on your campus.

Annual and Final Reports: Forms on which to submit annual and final reports on approved research projects are found on the REB Website. Annual reports are due on 15 January of each year, provided that this date is at least six months after the date of project approval. Final reports are due 90 days after project completion.

A Note about Review Procedures

Applications for research projects which involve no more than minimal risk of harm to research subjects, and which do not involve deception or incomplete disclosure, will normally receive delegated review. Projects ineligible for delegated review, or for which approval was not granted as a result of the delegated review process, are reviewed by the full REB. The status of your application at any time may be determined from the REB secretary on your campus.

FOR REB USE ONLY: **File No: _____**
Date Complete App'n. Rec'd: _____
Approved/Approved with: _____
Modification/Rejected: _____
Date: _____

……..

University of New Brunswick
Application for Review of Research Involving Humans

Principal Investigator(s): Name(s); Academic Status (Faculty, Undergraduate Student or Graduate Student); Academic Unit, e-mail Address, Office Telephone, Home Telephone:

Milton King; Graduate Student; Computer Science, milton.king@unb.ca, 902-447-2426
Paul Cook; Faculty; Computer Science, paul.cook@unb.ca, 506-447-3466

Title of Proposed Research:

Personalized Predominant Word Sense Identification

Commencement Date: **September 2020** Completion Date: **December 2020**

Co-Investigator(s): Academic Unit, e-mail address, Office Telephone

Supervisor(s) (if Principal Investigator is a student); Academic Unit, e-mail address, Office Telephone:

The undersigned parties certify that they have read, and undertake to comply fully with, the Tri-Council Policy Statement "Ethical Conduct for Research Involving Humans." (*Original signatures required*)

Principal Investigator(s): _____ Milton King ___Paul Cook

Co-Investigator(s): _____

Supervisor(s):_____

The undersigned certifies that the proposed research has been reviewed by, and is acceptable *in all respects* to, the academic unit(s) responsible. (*Original signature required*)

Dean/Director/Chair(s) Typed Name _____

Signature _____

Date submitted to the REB:

1. Summary: Provide here, in approximately 300 words, a summary of the proposed research, indicating clearly the role of the research subjects and any procedures to which they will be subjected.

Word sense disambiguation involves automatically assigning the correct meaning (also called sense) to a word within a context. For example, the word *bank* in *I need to deposit money at the bank* is most likely referring to the financial sense and not the sense that pertains to the side of a river. In this research, we propose to develop personalized models, i.e., models that have been tailored to a specific individual's writing, for word sense disambiguation and predominant sense identification. Currently, a sense annotated dataset does not exist for evaluating approaches to these tasks. We require annotators (research participants) to build this resource.

Using Amazon Mechanical Turk, participants will be shown sentences and a list of definitions for a selected word in each sentence. The participants will then be asked to select which of the definitions best fits the usage of the selected word in the sentence. An example of this annotation task is included below. The sentences are taken from existing and freely available datasets, i.e., the text does not come from the participants themselves. The annotated sentences will then be analyzed for inter-annotator agreement and used as a dataset for evaluating personalized approaches for automatic word sense disambiguation and predominant sense identification.

2. Risk: In your opinion, does this research pose more than minimal risk (Tri-Council Policy, Chapter 2, Section B) to participating subjects? ***No***

If yes, provide here a statement that describes in detail the aspects of the research procedure that pose a risk to subjects, and provide your assessment of the risk of harm (probability and severity). Note that not only physical injury but also anxiety or embarrassments are included in the concept of harm. Describe means adopted to minimize risk, and means (such as provision of counseling) to deal with harms, which subjects may experience. Describe as well the potential benefit, which will result from this research, which justifies the above risk of harm.

3. Deception: Does this research involve deception or partial disclosure? ***No***

If yes, refer to the Tri-Council Policy, Chapter 3, specifically Article 3.7 and subsequent commentary, and provide here an explanation of how you plan to comply with the requirements of that Section for debriefing. Describe as well the potential benefit, which will result from this research, which justifies waiving the normal requirements for full disclosure.

4. Funding: Has funding been received for this research? ***No***

If yes, ***from*** what agency and for what period?

5. Research Subjects:

5.1 Number of Subjects: How many subjects will participate in this research?

Our dataset will consist of approximately 2000 sentences which will be divided into groupings of roughly 5 sentences each, known as a HIT (Human Intelligence Task) on Amazon Mechanical Turk. We will therefore have approximately 400 HITs, each of which will be annotated by 5 participants. Participants will be able to choose how many HITs to complete. The total number of participants could therefore range from 5 (if all participants completed all 2000 HITs) to roughly 2000 (if each participant completed just one HIT).

5.2 Recruitment: How will they be recruited, and from what population?

Participants will be recruited through the online crowdsourcing service Amazon Mechanical Turk. Our study will be advertised on MTurk to participants who can then self-select to complete the task. The participants will be from either Canada or United States and must be able to speak English. Participants will be 19 years of age or older.

6. Informed Consent:

6.1 Informing Subjects: How will the nature of the research be explained to potential subjects, in compliance with Chapter 3, specifically article 3.2 and the subsequent commentary of the Tri-Council Policy? Attach a copy of any document(s), such as an explanatory letter, to be used for this purpose.

We include information about the task to potential participants prior to them performing the task as suggested by the guidelines on research use of crowd sourcing services from the University of Waterloo: https://uwaterloo.ca/research/office-research-ethics/research-human-participants/pre-submission-and-training/use-crowdsourcing-services. This information is included in the consent form (included below in 6.2).

Because the sentences to be annotated may include language that could be considered inappropriate or offensive, the task will be posted on Amazon Mechanical Turk with a title

that includes "WARNING: This HIT may contain adult content. Worker discretion is advised" as suggested by Amazon Mechanical Turk.

6.2 Consent: If written evidence of informed consent will be obtained, attach a copy of the consent form. (See Requirements for Informed Consent Forms.) **If written evidence of informed consent will not be used, explain here, in detail, how you intend to comply with the requirements of Chapter 3, Article 3.12 and the subsequent commentary of the Tri-Council Policy**:

The consent form is the first HIT, which participants are required to agree to in order to continue. It will contain the following information.

Title: Personalized Predominant Word Sense Identification
Researchers:

- **Milton King, PhD candidate, Faculty of Computer Science, University of New Brunswick, milton.king@unb.ca**
- **Dr. Paul Cook, Associate Professor, Faculty of Computer Science, University of New Brunswick, paul.cook@unb.ca, 1 506-447-3466**

Objectives of the research: Word sense disambiguation is a task that involves automatically determining the meanings of words. In this research, we are exploring the use of personalized methods in a word sense disambiguation task.

Procedures:

In the remainder of this HIT, you will be asked to select the definitions of highlighted words being used in sentences. This data will be used in academic research in natural language processing to evaluate computational models for automatically determining the meanings of words.

Potential Risks and Benefits: We do not foresee any risks for participators, except that they may be exposed to inappropriate text. For your participation, you will be rewarded monetarily for each HIT that you complete. We expect users to require approximately 1 minute per HIT.

Confidentiality: Confidentiality will be maintained throughout the research. No personal information will be collected other than your user ID.

Rights to Withdraw:

- **Your participation is voluntary. You may withdraw from the research project for any reason, at any time without explanation.**
- **Should you wish to withdraw, you may do so at any point.**
- **Your right to withdraw data from the study will apply for up to one week after completion.**

Questions or Concerns: Comments, concerns, questions and other feedback may be sent to any of the researchers listed above. Participants may also request the outcome of the research. If participants would like to contact the academics that are not involved with the

research, they can contact the Dean of Computer Science, Luigi Benedicenti, luigi.benedicenti@unb.ca.

Consent: I have read and understood the above information and do herby agree to participate in this study and consent to the use of the data recorded during this experiment for scientific research purposes stated above. I understand that my identity will remain confidential and that my user ID and results may be shared. I agree that all my questions and concerns have been addressed in a satisfactory manner and that I am of age 19 years or older.

6.3 Children as Research Subjects: If the proposed research involves children as subjects, provide here a statement indicating how compliance with Chapter 3, Section C, and specifically with Articles 3.9 and 3.10 of the Tri-Council Policy, will be achieved.

Not applicable.

6.4 Incompetent Adults as Research Subjects: If the research involves adults of diminished competence as subjects, provide a statement indicating how compliance with Chapter 3, Section C, and specifically with Articles 3.9 and 3.10 of the Tri-Council Policy, will be achieved.

Not applicable

7. Inducements: Will any inducements (money, grade points, etc.) be offered to encourage participation?

If yes, indicate here how compliance with Chapter 3, Article 3.1 and the subsequent commentary of the Tri-Council Policy (concerning voluntariness) will be achieved. If academic rewards are to be used, give details of alternative means of achieving equivalent rewards.

Yes, participants will be paid $0.05 per HIT. This amount is on par with the payment for similar tasks on Amazon Mechanical Turk, which follows the recommendations for rate of pay on crowd sourcing platforms in the guidelines on research use of crowd sourcing services from the University of Waterloo.

8. Private Information: Does the proposed research involve accessing identifiable personal information about subjects by means of surveys, questionnaires, etc.? ***No***

If yes, indicate here, in detail, how you propose to meet the requirements of the Tri-Council Policy, Chapter 5, Section B and C. A copy of any questionnaire, survey document or interview schedule to be used should be attached as well.

9. Feedback: Describe the measures, which you propose for providing feedback to research subjects concerning the outcome of the research.

In the consent form, participants are given contact information for the researchers, and the Dean of Computer Science, to be able to ask questions about the outcome of the research.

10. Data Security: Describe the measures, which you propose for ensuring the security of any identifiable personal data, which will be retained after completion of the research.

Personal information is not gathered from the participants, except for a unique participant id.

11. Continuing Review: All research requires brief annual reports and a brief report upon completion of the research. Suitable report forms are included at the end of this file. **Research involving more than minimal risk may require additional measures for continuing review.** If your research involves more than minimal risk, describe here the measures you propose for facilitating continuing review of this research, in compliance with Article 2.8 of the Tri-Council Policy.

A report will be filed with the REB upon the conclusion of the project. Adverse events will be reported to the REB and the project will be halted until the issue is found to have been adequately addressed.

12. Additional Information: Please feel free to append any additional information, which you feel may be helpful to the REB in evaluating this application.

N/A

Checklist for Attachments to Application for Review of Research Involving Humans

For items that are attached, indicate X; for items that are not applicable, indicate N/A.

Provide the following attachments where applicable:

X or N/A	
N/A	**Where the academic unit responsible for the research has a process of formal ethics review,** a copy of the approval notice from that process, together with any substantive comments provided by the reviewers.
N/A	**If external funding has been sought or obtained** for this research, **one copy only** of the complete application form as well as **two copies** of any reviewers' comments which have been received.
N/A	**For student research**, one copy of the full research proposal if one has been submitted to the relevant academic unit.
N/A	**A copy of any proposed information letter and/or informed consent form.** (Do not duplicate if already included in above material.)

X	**A copy of any questionnaire(s), survey documents or interview schedules** to be used in the research. (Do not duplicate if already included in above material.)
N/A	**A copy of any debriefing material to be provided to subjects.**
N/A	**For research under the jurisdiction of more than one institution**, an indication of which other REBs will review the research, and the results of such review if available (see Tri-Council Policy, Chapter 8, Section A).
N/A	**For all research (including student research) that exceeds minimal risk, which has not been approved by a sanctioned peer review process**, the applicant must recommend two reviewers competent to undertake a "scholarly review" of the proposed research. "Scholarly review" in this context refers to the process of determining whether the design of the research project is capable of addressing the questions being asked in the research.
N/A	**In all cases,** a full description of the proposed research, if this is not already contained in the material listed above.
X	**In all cases,** an electronic text version of the Summary (Item 1 on the application form), via e-mail to ethics@unb.ca on the UNBF campus or to reb@unb.ca on the UNBSJ campus.

Please append this checklist to the application form.

REQUIREMENTS FOR INFORMED CONSENT FORMS

DO NOT ATTACH THIS DOCUMENT TO YOUR APPLICATION

The purpose of an Informed Consent Form is to help to inform the potential research subject about the research and to document the subject's informed agreement to participate. One copy should be given to the research subject, and a copy signed by the subject should be retained in the researcher's files.

An Informed Consent Form must be written in language that the potential subject can understand. It must contain at least the following information:

- A statement identifying the researcher(s). This statement should indicate their affiliation with the University of New Brunswick, and how they may be contacted.
- For student research, the name, affiliation and telephone number of the student's supervisor(s).
- A statement identifying a person not directly involved in the research, for example the Department Chair or Faculty Dean, who may be contacted should the subject have concerns about the research.
- A statement that the reader is being invited to participate in research.

- A clear statement of the purpose of the research.
- A full description of the procedures to be followed in the research (e.g. what is expected of participants, what they will be asked to do, what data will be collected).
- The period of time required for subject participation.
- A statement indicating that participation is voluntary, and that subjects are free to withdraw from the research, and to withdraw any data pertaining to themselves, at any time, without penalty.
- A statement indicating how subjects may receive information as to the outcome of the research.
- Where applicable, a clear description of any potential discomforts and/or risks associated with participation in the research.
- Where applicable, a clear description of the potential benefits of the research, whether to the subject or to others.
- Where applicable, a statement concerning recording of subject participation (audio tape, video tape, photographic records, electronic data recordings, etc.); who will have access to the records, security provisions in storage, possible use in publication, and when they will be erased or destroyed.
- Where applicable, a statement concerning compensation for any injury incurred while participating in the research.
- Where applicable, (for any research involving questionnaires or interviews), a statement informing subjects that they may decline to answer specific questions.
- Where applicable, a statement indicating how confidentiality will be protected.
- Where applicable, a clear explanation of any inducements offered for participation, and of the consequences (if any) on those inducements if the subject withdraws before the research is completed.

Children as Research Subjects:

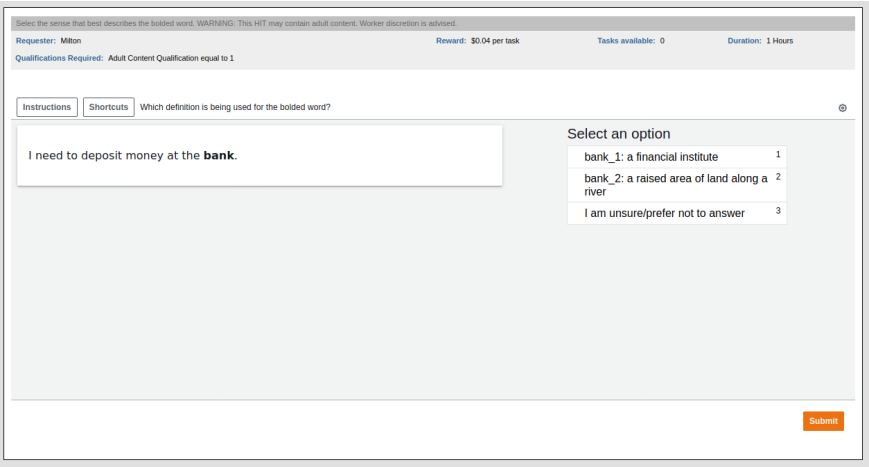
A parent or legal guardian must provide signed written consent for the participation of children in research. As well, assent of the child is normally required where the child is old enough to understand, and evident dissent is always an indication that the child's participation in the research should be terminated.

When participants under nineteen are considered by the applicant to be competent to consent without parental involvement, the applicant must provide justification for this conclusion.

Incompetent Adults as Research Subjects:

For research involving adults of limited competence, signed written consent must be provided by an authorized third party. As well, assent of the subject is normally required, and evident dissent is always an indication that participation in the research should be terminated.

(Sample consent forms consistent with the requirements of the TCPS Policy exist on the Queen's University website: http://www.queensu.ca/ors/researchethics/GeneralREB/forms/loiancf.html)



An example of the annotation task for a sentence including the word *bank*.

A.2 Author-level Dataset Statistics

Information about authors included in the proposed word sense disambiguation dataset is shown in Table A.1.

A.3 Author-level Sense Distributions

Author-level sense distributions for each lemma in my proposed word sense disambiguation dataset are shown in Figures A.1 - A.11. The average distributions across all authors is in black.

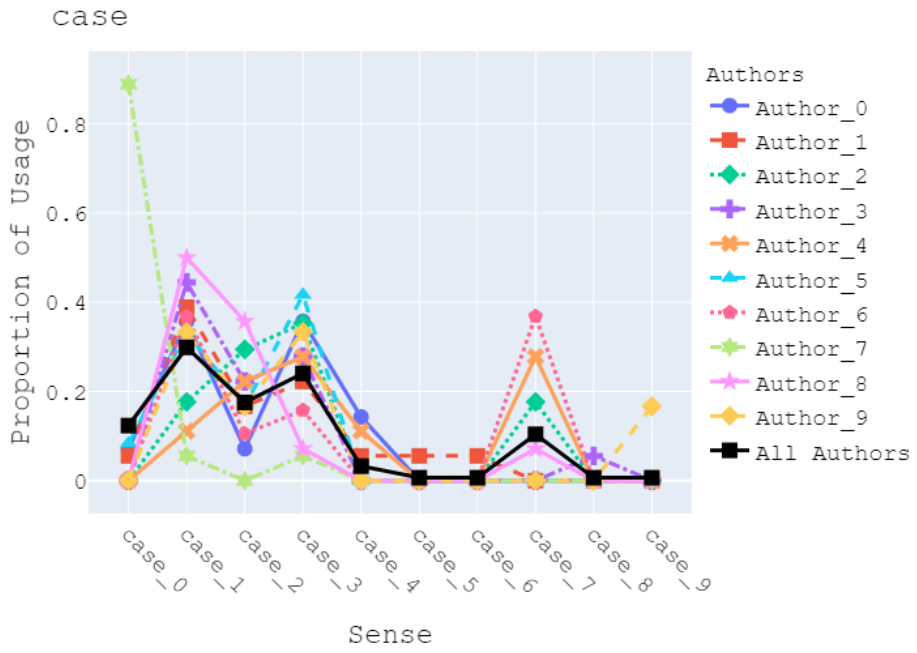


Figure A.1: Sense distributions for *case*.

ID	# Instances	Age	Sex
780903	152	25	male
1476382	144	33	male
449628	112	34	male
1862467	96	27	female
942828	87	34	female
1151815	69	25	male
1046946	62	25	female
665500	60	35	male
554681	59	45	female
595404	58	48	female
1662633	53	23	female
122217	52	37	male
2186817	50	36	male
1713442	48	34	male
883178	41	36	male
1234212	39	27	male
861706	39	24	male
605396	32	35	male
1889734	31	25	male
216413	30	26	female
1281160	29	24	male
470861	29	27	male
2922061	28	44	male
1516660	24	17	male
1650898	22	38	male
1013637	20	17	male
1268682	20	35	male
664485	20	25	male
681976	13	26	female
3308548	12	26	male
546850	12	24	male
49663	12	33	male
3858248	11	27	male
1325355	9	26	female
2994939	8	37	male
619488	3	25	female
Min	3	17	-
Max	152	48	-
Mean	44	30	-
Median	31	27	-

Table A.1: The details of the dataset with respect to the number of instances per author in the dataset, the age of the authors, and the sex of the authors.

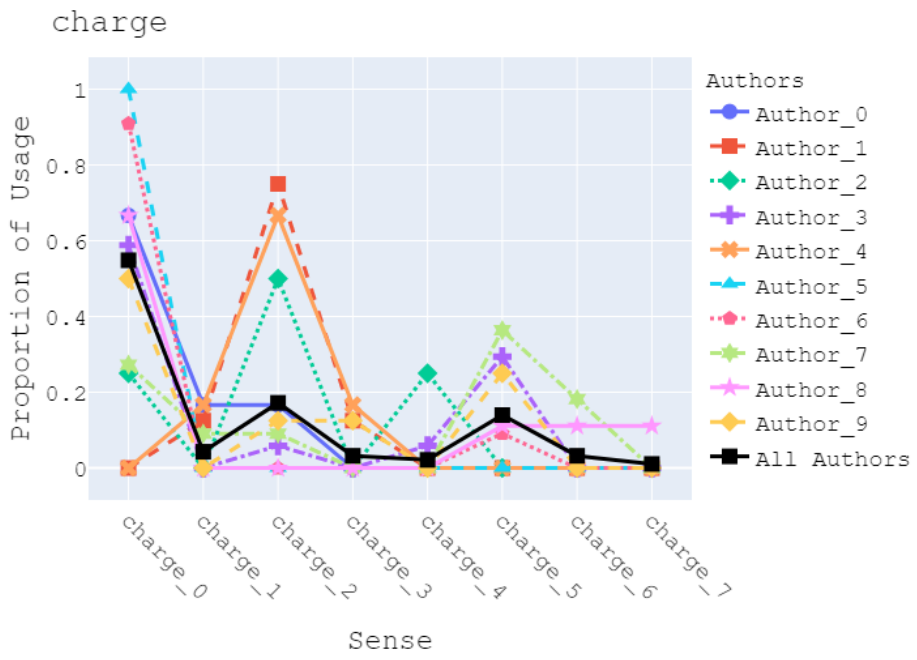


Figure A.2: Sense distributions for *charge*.

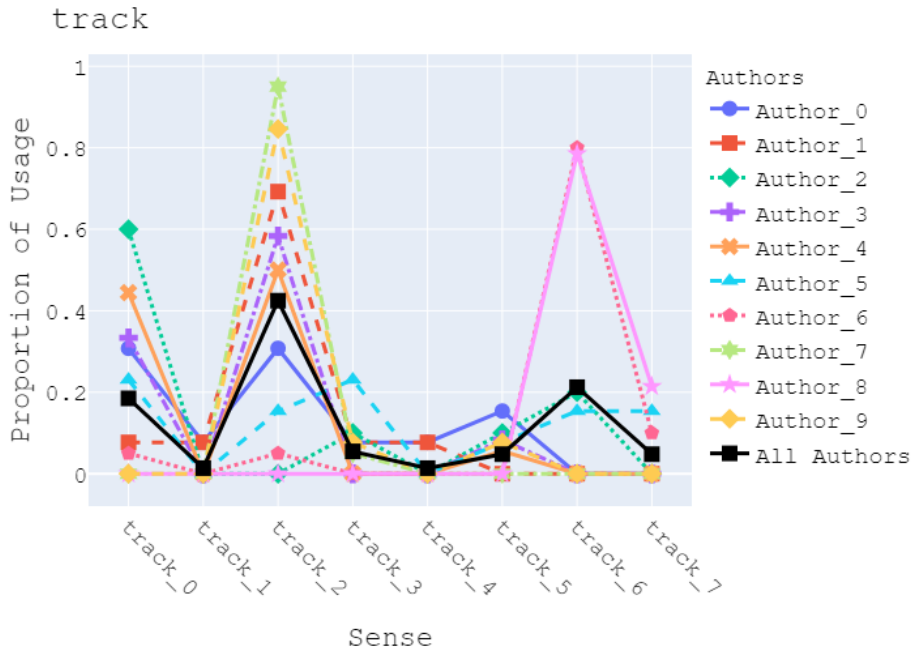


Figure A.3: Sense distributions for *track*.

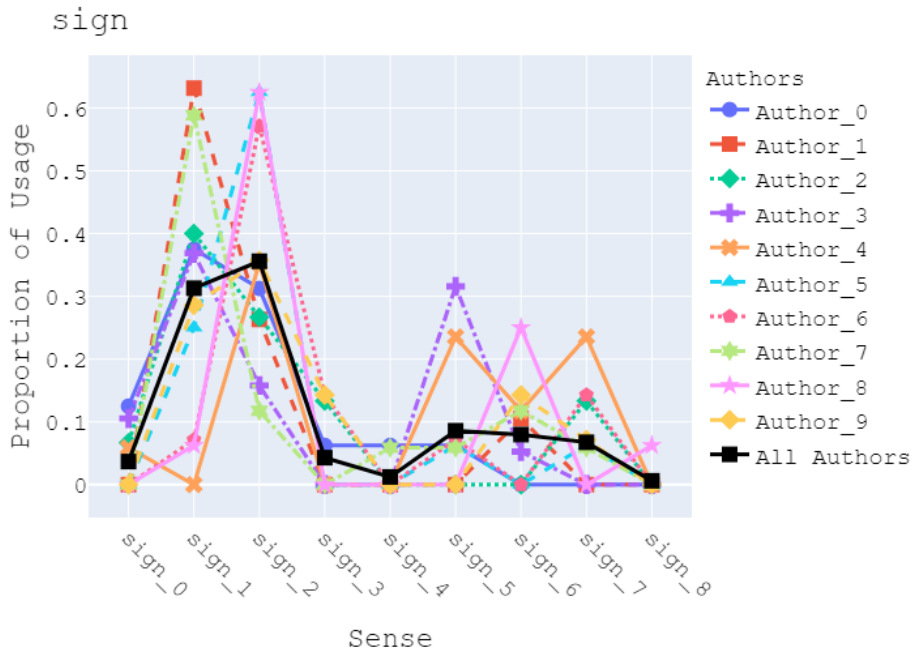


Figure A.4: Sense distributions for *sign*.

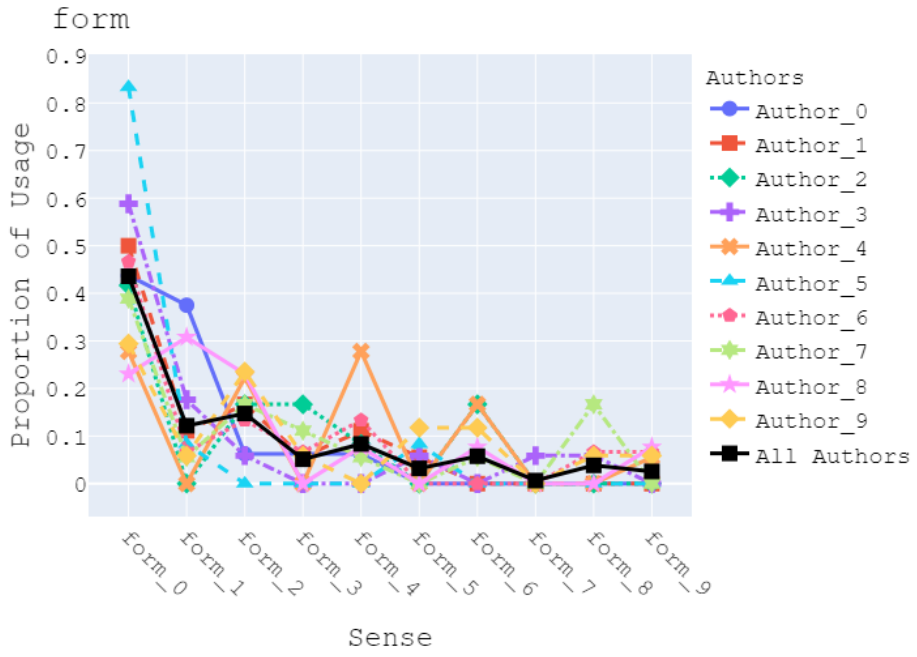


Figure A.5: Sense distributions for *form*.

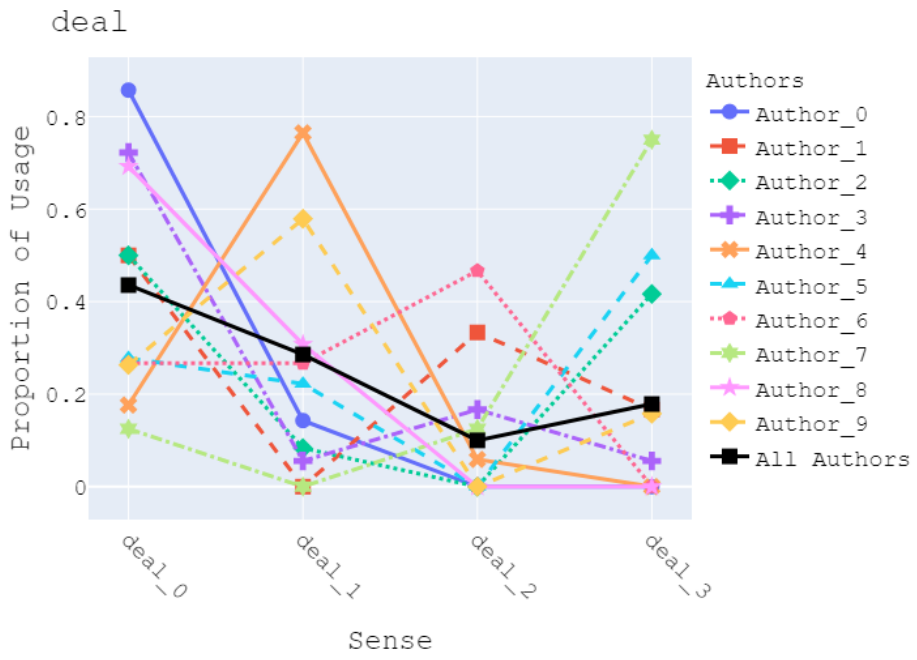


Figure A.6: Sense distributions for *deal*.

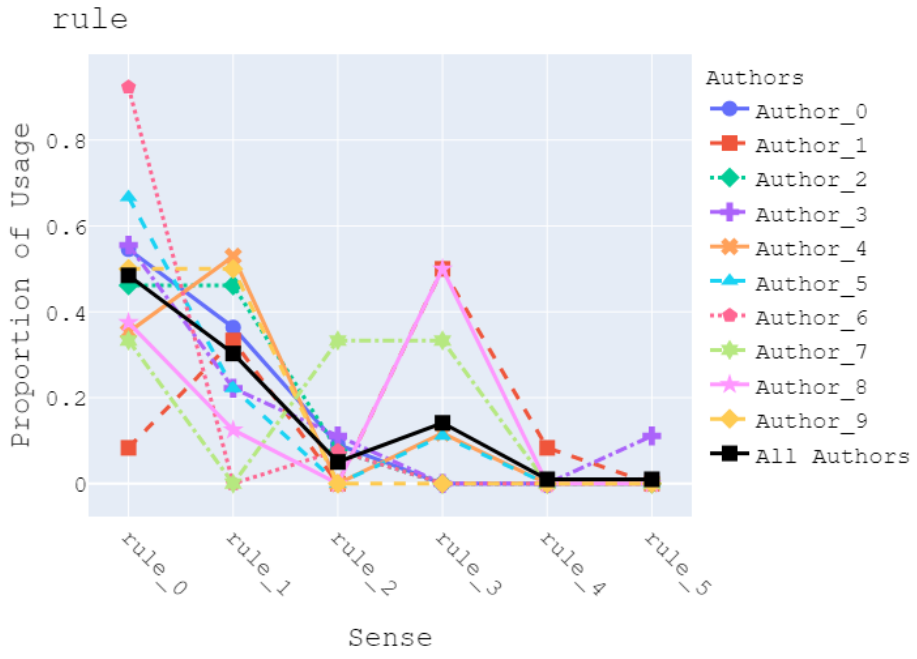


Figure A.7: Sense distributions for *rule*.

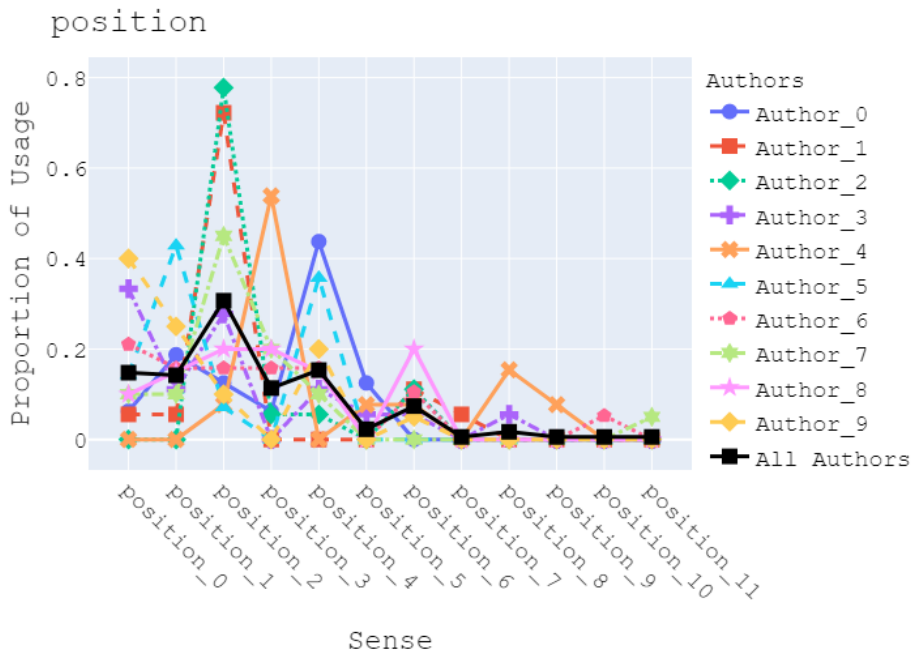


Figure A.8: Sense distributions for *position*.

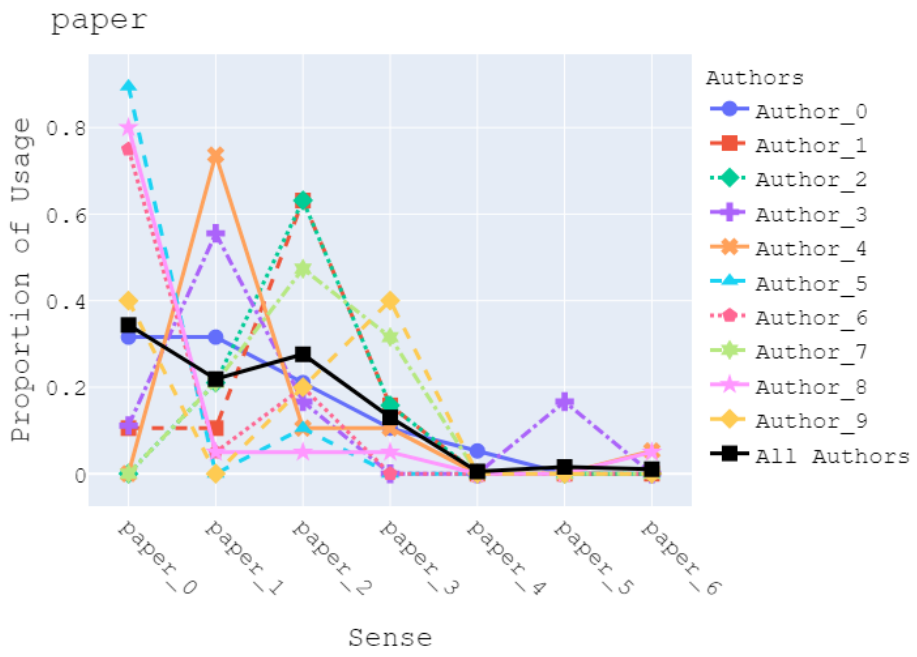


Figure A.9: Sense distributions for *paper*.

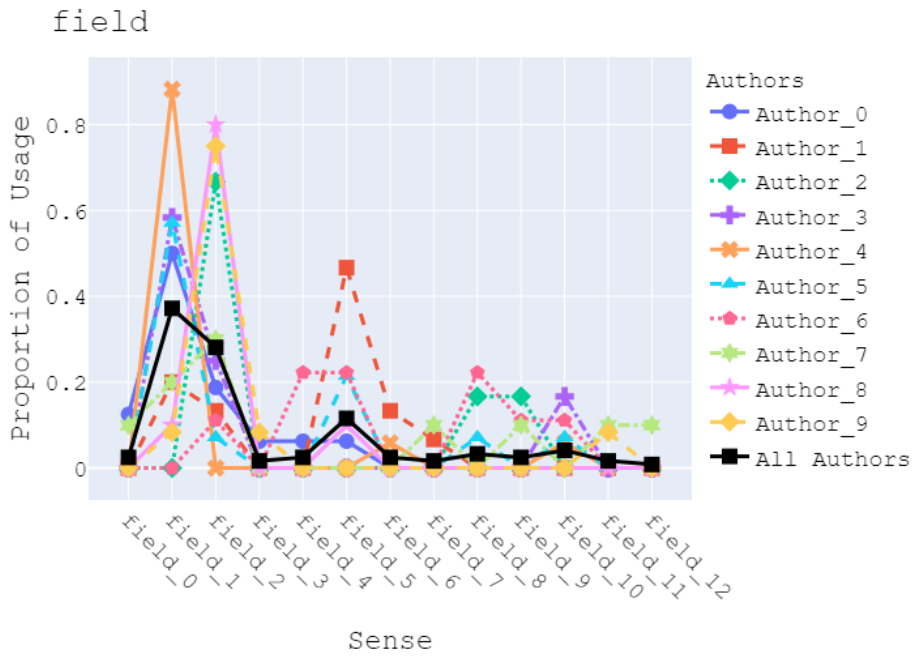


Figure A.10: Sense distributions for *field*.

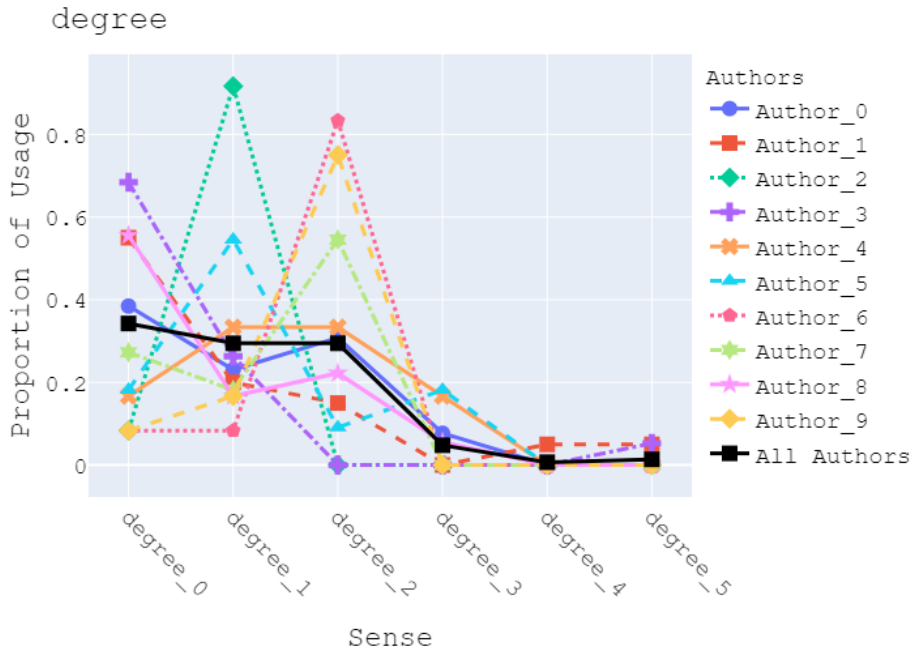


Figure A.11: Sense distributions for *degree*.

Curriculum Vitae

Candidate's full name: Milton King

Universities attended:
University of New Brunswick, 2015-2017, Master's in Computer Science.
Mount Allison University, 2011-2015, Bachelor of Science.

Peer-reviewed conference papers:

Milton King and Paul Cook. 2021. Now, It's Personal: The Need for Personalized Word Sense Disambiguation. In Proceedings of The Recent Advances in Natural Language Processing 2021 (RANLP), pages 697–705. Online.

Milton King and Paul Cook. 2020. Authorship Verification with Personalized Language Models. In: Sojka P., Kopeček I., Pala K., Horák A. (eds) Text, Speech, and Dialogue. TSD 2020. Lecture Notes in Computer Science, vol 12284, pages 248–256. Springer, Cham.

Milton King and Paul Cook. 2020. Evaluating Approaches to Personalizing Language Models. In Proceedings of The 12th Language Resources and Evaluation Conference, pages 2461–2469. Marseille, France.

Milton King and Paul Cook. 2019. Building Personalized Language Models through Language Model Interpolation. In Proceedings of the 20th International Conference on Computational Linguistics and Intelligent Text Processing (CICLing 2019). La Rochelle, France. (To appear in Springer Lecture Notes in Computer Science)

Milton King and Paul Cook. 2018. Leveraging distributed representations and lexico-syntactic fixedness for token-level prediction of the idiomaticity of English verb-noun combinations. In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), pages 345–350. Melbourne, Australia.

Milton King, Dima Alhadidi and Paul Cook. 2018. Text-based Detection of Unauthorized Users of Social Media Accounts. In Bagheri E., Cheung J. (eds) Advances in Artificial Intelligence. Canadian AI 2018. Lecture Notes in Computer Science, vol 10832. Springer.

Peer-reviewed workshop papers:

Milton King, Ali Hakimi Parizi, Samin Fakharian, and Paul Cook. 2021. UNBNLP at SemEval-2021 Task 1: Predicting lexical complexity with masked language models and character-level encoders. To Appear In Proceedings of the 15th International Workshop on Semantic Evaluation (SemEval-2021). Bangkok, Thailand.

Ali Hakimi Parizi, Milton King and Paul Cook. 2019. UNBNLP at SemEval-2019 Task 5 and 6: Using Language Models to Detect Hate Speech and Offensive Language. In Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval-2019), pages 514–518. Minneapolis, USA.

Milton King, Ali Hakimi Parizi and Paul Cook. 2018. UNBNLP at SemEval-2018 Task 10: Evaluating unsupervised approaches to capturing discriminative attributes. In Proceedings of The 12th International Workshop on Semantic Evaluation (SemEval 2018), pages 1013–1016. New Orleans, Louisiana.

Milton King and Paul Cook. 2017. Supervised and unsupervised approaches to measuring usage similarity. In Proceedings of the 1st Workshop on Sense, Concept and Entity Representations and their Applications, pages 47–52. Valencia, Spain.

Milton King, Waseem Gharbieh, SoHyun Park and Paul Cook. 2016. UNBNLP at SemEval-2016 Task 1: Semantic Textual Similarity: A Unified Framework for Semantic Processing and Evaluation. In Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016), pages 732–735. San Diego, United States.

Presentations:
2020. Evaluating Approaches to Personalizing Language Models. The 12th Language Resources and Evaluation Conference, Marseille, France. Online.

2019. Building Personalized Language Models through Language Model Interpolation. The 20th International Conference on Computational Linguistics and Intelligent Text Processing (CICLing 2019). La Rochelle, France.

2019. Building Personalized Language Models Through Language Model Interpolation. The 43rd Annual Meeting of the Atlantic Provinces Linguistics Association. Fredericton, New Brunswick.

2018. Leveraging distributed representations and lexico-syntactic fixedness for token-level prediction of the idiomaticity of English verb-noun combinations. The 56th Annual Meeting of the Association for Computational Linguistics. Melbourne, Australia.

2018. Text-based Detection of Unauthorized Users of Social Media Accounts. Canadian AI 2018. Toronto, Canada.

2017. Supervised and unsupervised approaches to measuring usage similarity. The 1st Workshop on Sense, Concept and Entity Representations and their Applications. Valencia, Spain.

2017. Determining if this word is used like that word: Predicting usage similarity with supervised and unsupervised approaches. Science Atlantic Mathematics, Statistics and Computer Science Conference 2017.

2016. Supervised and unsupervised approaches to measuring usage similarity. Science Atlantic Mathematics, Statistics and Computer Science Conference 2016.

Theses:
Milton King. 2017. Determining if this Word is Used like That Word: Predicting Usage Similarity with supervised and unsupervised approaches. MCS. thesis, Faculty of Computer Science, University of New Brunswick.