Content and Stylistic Models for Authorship, Stance, and Hyperpartisan Detection

by
Marjan Hosseinia

A dissertation submitted to the Department of Computer Science,

College of Natural Sciences and Mathematics

in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in Computer Science

Chair of Committee: Dr. Arjun Mukherjee

Committee Member: Dr. Ricardo Vilalta

Committee Member: Dr. Omprakash Gnawali

Committee Member: Dr. Xuqing Wu

University of Houston
May 2020

ACKNOWLEDGMENTS

I would like to thank everyone who made an important contribution to this project. My advisor, Dr. Arjun Mukherjee, deserves special praise for being an invaluable mentor to guide me in the right direction. I am also grateful to my other Ph.D. committee members, Dr. Ricardo Vilalta, Dr. Omprakash Gnawali, and Dr. Xuqing Wu for sharing their knowledge and experience and for taking the time to serve on my dissertation committee. I would like to thank my labmates and friends for their support during the last five years: Yifan Zhang, Fan Yang, and Santosh K.C. My greatest thanks go to my family for the endless support throughout this process. Your love and care have always guided me to succeed.

ABSTRACT

This dissertation presents content and stylistic solutions for three opinion-oriented text classification problems. It explores user-generated text data to find how individuals write through authorship identification, express their opinion via stance detection, and articulate news while belonging to the left or right political party using hyperpartisan news detection.

In the first problem, this research studies the case of deception detection in online reviews. It compares the distribution of structural features of the text using KL-Divergence to find the most discriminative elements of an individual's writing style. Then, it proposes a transductive algorithm to learn from unlabeled test data to expand the training set. Following that, it focuses on authorship verification for document pairs with different topics, genres, or both by presenting a neural network model with parallel recurrent layers and a fusion mechanism that compares the language of the two documents. The model is examined on datasets of multiple domains, including multi-topics multi-genre PAN datasets, Amazon reviews, and a dataset of machine learning articles. According to the experimental results, the model achieves stable and competitive performance compared to the baselines. Finally, a hierarchical version of the network with two layers of attention is designed for detecting writing style change within a text document. The model takes the structural features of a sentence to observe the transitions of writing style. Experimental evaluation on the PAN 2018 dataset confirms our previous finding of the effectiveness of structural elements in representing writing style.

In the second problem, this research works on identifying the stance of argumentative opinion, a novel application of opinion mining. Its proposed data consists of arguments represented in nonpartisan format. While it is acknowledged that accurate information from both sides of the contemporary issues is an 'antidote to confirmation bias' and such information helps the society to improve critical thinking and open-mindedness, it is relatively rare and hard to find online. With the well-researched non-biased arguments on controversial issues shared by Procon.org, detecting the stance of arguments is a crucial step to automate organizing such resources. To address this, it employs a universal pretrained language model with a weight-dropped LSTM neural network to leverage the context of an argument for finding the argument's stance. The analysis shows the

strength of pretraining and the ability of the model to find the stance of long arguments through the entire documents using pooling operations.

Finally, this dissertation provides an approach to see if the latent personality features in individuals' writing can be useful in the three opinion-oriented classification tasks. The approach deploys the state-of-the-art deep bidirectional transformer to extract the Myers-Briggs personality type from user posts. The posts are collected from Reddit, Twitter, and a personality forum with the self-reported personality type by the users. Then, it induces personality information from its proposed transformer-based model and combines the information with some other classification models. Experimental evidence shows the effectiveness of personality information in authorship verification, stance detection of arguments, and hyperpartisan news detection after topic-based sub-sampling of the news training data.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1 Introduction

From the very beginning time of scripts in ancient Sumer to the current century, written materials have been used to express human's beliefs, views, findings and as a tool for communication. Today, however, technology has made the written materials available around the globe through web platforms. As we speak the amount of online text data is increasing dramatically. Internet users are able to express their voice through online forums, social media platforms, and blogs effortlessly. Formal contents from scientific publications, to the news articles and books are accessible through online resources. Despite the enormous advantages of the ease of sharing text data in various forms and its availability, it can cause serious threats to the public.

User posts and online reviews on Web resources are among the main publicly available data types. Studies show that 80% of Americans use some sort of social media platforms and 90% of people read online reviews before they shop. However, the popularity of these types of data makes them a suitable choice to be misused.

Recently, fake identities have been created by people to deceive shoppers through reviews for or against a product. These misleading information can directly impact the reliability of online reviews, consumerism, and e-commerce. While there are some efforts that identify such users based on meta information including user IP or time of posts, it is still essential to find the relationship of suspicious reviews based on the contents they provide and the way they are written. While authors may bypass security checks by changing their IPs or other settings, it is not always easy for them to change their intrinsic features appearing in their reviews while giving persuasive opinions. So, if a group of reviews share an acceptable amount of content and stylistic features it is probable that they are from one single author.

Aside from online reviews, it is essential to investigate the authorship of text documents for some other real world scenarios such as legal writing analysis (e.g., forensic analysis and copyright infringement), plagiarism detection, and for enhancing the security of online systems by tracking their users based on the contents that they provide. Although there might be no or a few resources

available for some specific applications, an authorship identification system trained on different types of user generated text can still contribute to the scenarios with no or few resources. In another aspect, opinions shared online have persuasive power and can form public beliefs and promote confirmation bias. Confirmation bias, the humans' tendency to reinforce their existing beliefs permeates the internet debates and specifically hyperpartisan news articles, highly skewed opinions to one political party. The high impact of such resources target their audience and can promote misinformation, hate speech, and discrimination. Many systems are being designed to identify and remove such contents in online platforms shared by their users. However, improving solutions to avoid confirmation bias from human's perspective has been less studied and is essential to protect society. In this aspect two strategies can be taken. First, natural language processing models can identify hyperpartisan news articles to help the reader not to consume the hyperpartisan contents unintentionally. In another way, online resources with accurate argumentative opinions represented in non-partisan format should be improved. Such resources promote critical thinking and open mindedness and are an antidote to confirmation bias. In this regard, identifying the stance of such arguments towards an issue is a crucial step to automate this process. The stance reveals one's posture towards a topic that can be pro or con the topic in a binary setting.

With the importance of the aforementioned problems emerged with the availability of online text data, this work explores the content and stylistic solutions for the three opinion-oriented text classification tasks, including authorship identification, stance detection, and hyperpartisan news detection. Each of these tasks is designed around opinion mining and connects with opinion. In authorship identification, we investigate the way individuals write and express their opinion to address two important questions. First, whether two pieces of text are written by the same person. Secondly, whether a text document is authored by one single person at the first place. We study this problem on various domains including reviews, user posts, essays and scientific articles. In stance detection, we find the stance of argumentative opinion against a contemporary issue. These arguments are provided by Procon.org, a non-profit organization in non-partisan format. Finally, in hyperpartisan news detection, we investigate how people express their opinion while articulating

one-sided news articles for two available hyperpartisan news datasets. In the following we discuss our contributions in each problem.

1.1 Dissertation Contributions

We start with sockpuppets detection in deceptive opinion spam. While detecting deception opinions has been researched from both linguistic and behavioral aspects, the case of sockpuppets has remained unsolved. A sockpuppet refers to a physical author using multiple user-ids to deliver fake reviews to avoid getting filtered. These reviews, whether positive or negative, influence the public's view and impact a business's performance. While sockpuppets can change their IP address or exercise various profiles, it is not easy for them to alter their writing style when expressing an opinion. We seek the writing style as the author's fingerprint in the reviews to identify sockpuppets. The choice of vocabulary and the grammar that individuals use for structuring their sentences shape their writing style. So, by modeling the way individuals write, we can identify them using their generated text. We first, create a sockpuppet dataset using Amazon Mechanical Turks. Then, we propose a feature selection method using KL-Divergence that finds the most discriminative grammatical features of a sentence in the authors' writing. Finally, we design a spy-Induction method that leverages the unlabeled test data during the training phase to improve the diversity of authors and the accuracy of the model [37].

Following that, this research studies the Authorship Verification (AV) problem for a real-world scenario on various types of data, including scientific articles, online reviews, short essays, and novels originated from different genres, topics, or both. The AV method identifies whether two pieces of writing have the same author or not. In most previous works on AV, a large number of writing samples of a few known authors were given, and a verifier was designed for each author. However, these conditions are far beyond real-world scenario where the authors are unknown to us, the text documents are as short as a few sentences, and they can even be from two different domains. We propose a parallel neural network architecture that compares the language of the two input documents using several vector similarity measures. Evaluation results on experimental datasets

represent that the model provides sufficient linguistic features to verify the authorship reliably. To interpret the deployed techniques, we plot 2D projections of the fusion layer of the same and different-authorship input pairs. The plots confirm the ability of the model to differentiate the two classes using its adopted techniques [38].

While authorship identification methods usually assume that one single person authors the underlying documents, it is not valid in a more realistic case. In the last part of this task, we work on the new writing style change detection to first check whether the documents have one or multiple authors. This problem has been recently introduced by PAN organizers. Following our two previous approaches, we improve the parallel neural network architecture by adding a feature-based layer, a sentence-based layer, and two layers of attention. Our proposed model utilizes the hierarchical structure of sentences derived from a statistical grammar parser in its input as crucial features to detect any writing style change within a document [39].

Next, this research studies the way individuals take a stance about an issue when expressing their argumentative opinions. Recently, the argumentative opinions of controversial issues have attracted more critical thinkers who want to take a stance after seeking enough information about the reason behind opinions from both sides. Most works on stance classification focus on online debates, including posts and tweets, while typically restricting the underlying data to a few (up to 8) targets (issues). While the works in stance detection have made essential contributions, they do not address the problem of detecting stance in the fluid and long arguments of a wide range of issues, which is the focus of our second task. To this end, we propose a new stance detection dataset from ProCon[1], a collection of critical, controversial issues. We, then, propose a recurrent neural network model that leverages the context of an issue using a pre-trained language model to predict the stance of the given argument against one of the 46 different issues. We extend our work by proposing the new version of the Procon dataset with 419 different controversial (sub-)issues and engaging VADER sentiment to the BERT model for stance detection.

Finally, this research proposes a general approach for all three opinion-oriented tasks with more

[1]https://www.procon.org/

focus on hyperpartisan news detection. The importance of detecting hyperpartisanship in the news has dramatically increased after the 2016 presidential election, and some resources have identified the skewness of news articles manually. However, the high speed of news release requires an efficient automatic solution that detects the status of hyperpartisanship of articles. As this problem is in its early steps, and a news article is a formal representation of an author's opinion, we aim to work on it from a new perspective. Our approach is to seek out latent features associated with the author's personality other than regular linguistic features in his/her writing. Accordingly, we extract Myers-Briggs personality types from an individual's writings to find a connection between personality type and opinion. First, we collect a unified dataset of three personality datasets, each of which consists of user posts of a social media platform with self-reported Myers-Briggs personality type. Then, we design a new version of the state-of-the-art bidirectional transformer by engaging the flow of sentiment of sentences to predict personality types in the multi-label classification scheme. The trained personality model is used to derive the personality information of the author from his/her writings to improve the tasks designed around opinions. As a further step, we connect the personality model with BERT for authorship verification, stance detection, and hyperpartisan news detection. We also provide analysis to show under which circumstances the proposed personality fingerprint would be practical.

1.2 Structure of the Dissertation

The rest of the dissertation is organized as follows. Chapter 2 provides a background overview and explores the most relevant works to the three underlying tasks. The focus of Chapter 3 is on our proposed models for sockpuppet detection, authorship verification, and writing style change detection, all of which are applications of authorship identification. In Chapter 4, we propose a new dataset and a neural network model for stance detection of argumentative opinions. Finally, Chapter 5 presents our general approach based on personality types for the three underlying tasks in this dissertation, followed by a discussion of main ideas and future works in Chapter 6.

2 LITERATURE REVIEW

This chapter provides an overview of the most relevant research on the specified opinion-oriented tasks.

2.1 Authorship Verification

In Authorship Verification, given the writings of an author, the task is to determine if a new document is written by that author. Koppel and Schler, (2004) explored the problem on American novelists using one-class classification and the "unmasking" technique [53]. Unmasking exploits the rate of deterioration of the accuracy of learned models as the best features are iteratively dropped. In [54], the task was to determine whether the same author wrote a pair of blogs. Repeated feature sub-sampling was used to determine if one document of the pair allowed selecting the other among a background set of "imposters" reliably. Effective unmasking requires a few hundred words to gain statistical robustness and was shown to be ineffective for short texts (e.g., reviews) in [99].

In the majority of AV approaches the authors are known to us, and a verifier trains the language model of the future authors [45], [53], [67], [37]. However, in a more difficult case **no** writing samples of a questioned author are specified, and they are unknown to us. No general solution has been offered for the verification problem under this assumption till 2014 [54]. Since then, a few works exist in the literature: Koppel and Winter [54] propose an almost unsupervised method for the blog corpus dataset using the "impostors" method. Optimized Classification Trees, the winner method of the PAN2014 Essays dataset, optimizes a decision tree based on various types of features and different comparison methods, including cosine similarity, correlation coefficient, and Euclidean distance [25]. Multi-headed RNN is a character-level RNN and contains a common recurrent state among all authors with an independent softmax output per author [6]. Fuzzy C-Means clustering, the winner of the PAN2014 competition for novels dataset, adopts C-Means clustering and lexical features for the task [74]. Recently, an approach based on the compression models has been evaluated on PAN datasets [33]. It achieves promising results for the two years

of PAN competitions but not for the other two datasets. Our method is similar to these methods and considers the problems with the binary structure, but we examine them on all PAN small-scale datasets as well as two large scale datasets.

2.2 Authorship Attribution

Authorship Attribution solves the attribution problem on a closed set of authors using text categorization. Supervised multi-class classification algorithms with lexical, semantic, syntactic, stylistic, and character n-gram features have been explored in [32, 28, 100]. In [96], a tri-training method was proposed to solve AA under limited training data that extended co-training using three views: lexical, character, and syntactic. The method, however, assumes that a large set of unlabeled documents authored by the same given closed set of authors is available, which is different from our sockpuppet verification. In [101], latent topic features were used to improve attribution. This method also requires larger text collection per author to discover the latent topics for each author which is unavailable for a sockpuppet.

2.3 Sockpuppet Detection

Sockpuppets were studied in [104] for detecting fake identities in Wikipedia content providers using an SVM model with word and Part Of Speech (POS) features. In [95], a similarity space-based learning method was proposed for identifying multiple user-ids of the same author. These methods assume a reasonable context (e.g., 30 reviews per user-id). These may not be realistic in opinion spamming (e.g., [79, 46, 27]) as the reviews per user-id are far less and often only one, as shown in singleton opinion spamming [117].

2.4 Stance Detection

In terms of underlying data, most works on stance classification focus on online debates, including posts and tweets, while typically restricting the underlying data to a few (up to 8) targets (issues). A probabilistic approach that models stance, the target of stance, and sentiment of a tweet is trained

on a large dataset of around 3K and evaluated on more than 1.2K tweets toward five targets [21]. Hasan et al. explore a new task of reason classification by modeling stances and reasons on a corpus of ideological debate posts from five domains [35]. They show that developed models of stances and reasons provide better reason and stance classification results than their simpler models. SemEval 2016 provides two different stance detection frameworks for tweets [75, 20]: one supervised framework with five targets containing 4K tweets and another weakly supervised with one target [5]. Tweets are also studied in some other works for stance detection [52, 13, 65, 103].

Unlike others, Bar et al. propose a contrast detection algorithm for 55 different topics [8]. Their algorithm is designed to detect the stance of *claims* that are defined as *"general, concise statements that directly support or contest the given topic"*. Under their definition, claims are *"often only a small part of a single Wikipedia sentence"* in their dataset [58]. The algorithm is based on hand-crafted lexicons and assumes critical phrases in the claims are already recognized.

In terms of features, the works are mostly based on the use of linguistic features, such as word n-grams, parse trees, opinion lexicons, and sentiment to determine the stance of an opinion concerning a controversial topic [106, 3, 107, 34, 107, 112].

Most recently, stance detection models employ different variations of neural networks, including memory networks [77], recurrent neural networks [5] or attention mechanism [20], however, none of them have acquired language models from a massive external resource.

While all these works have made important contributions, they do not address the problem of detecting stance in fluid and long arguments, which is the focus of this work.

2.5 Hyperpartisan Detection

Partisanship is the quality or action of strongly supporting a person, principle, or political party often without considering or judging the matter very carefully. Recently, the term hyperpartisan news is being used to define the extremely biased news in favor of the two right and left political parties. The importance of detecting the hyperpartisanship in the news has dramatically increased after the 2016 presidential election, and some resources have identified the skewness of news articles

manually. Hyperpartisan news detection is defined as a shared task in SemEval 2019. The submitted systems to the shared task use a wide variety of features from typical word n-grams of the bag-of-words model and various hand-crafted lexicons to meta HTML-based features that identify the target of hyperlinks [49]. The complete list of features used in hyperpartisan news detection is organized as follows.

- Word, character, and part-of-speech $n-$grams are the main features of many bag-of-words systems.
- Token embeddings, including Word2Vec, fastText, GloVe, ELMo, and state-of-the-art BERT, are considered in both neural network and classical models.
- Stylometric features such as punctuation, article structure, readability scores, psycholinguistic dictionaries have been added as auxiliary features to the models. Moreover, a hand-crafted profanity lexicon is created due to the occurrence of profanities more often in hyperpartisan contents.
- Polarity features gathered from built-in sentiment and emotion libraries or dictionaries are used in the different systems.
- Named entity types are extracted from text in one system, but only the type of "nationalities or religious and political groups" was found to be effective.
- Quotations are considered separately as features in some works but are removed from text in some other works.
- Targets of hyperlinks are divided into partisan and non-partisan sources in some works. It is assumed that the hyperpartisan articles tend to cite the resources from the same political party like theirs.
- Meta information such as date of publication is indicated as a sign of partisanship in some works. It is assumed that partisanship was more common at the time of the 2016 US presidential election.

Taking into account the contributions of the above approaches, none of them studies the effect of opinion-oriented features associated with the personality of the author that is the focus of our work in hyperpartisanship.

In the next three chapters we discuss our proposed models and contributions for the three underlying problems in details.

3 AUTHORSHIP IDENTIFICATION

We study the three important tasks in the area of authorship identification I) deception detection, II) authorship verification and II) writing style change detection. We explain the details of each study in the following sections.

3.1 Deception Detection

Deceptive opinion spam refers to illegitimate activities, such as writing fake reviews, giving fake ratings, etc., to mislead consumers. While the problem has been researched from both linguistic [83, 24] and behavioral [78, 62] aspects, the case of sockpuppets still remains unsolved. A sockpuppet refers to a physical author using multiple aliases (user-ids) to inflict opinion spam to avoid getting filtered. Sockpuppets are particularly difficult to detect by existing opinion spam detection methods as a sockpuppet invariably uses a user-id only a few times (often once) thereby limiting context per user-id. Deceptive sockpuppets may thus be considered as a new frontier of attacks in opinion spam.

However, specific behavioral techniques such as Internet Protocol (IP) and session logs based detection in [59] and group spammer detection in [79] can provide important signals to probe into few ids that form a potential sockpuppet. Particularly, some strong signals such as using the same IP and session logs, abnormal keystroke similarities, etc. (all of which are almost always available to a website administrator) can render decent confidence that some reviews are written by one author masked behind a sockpuppet. This can render a form of "training data" for identifying that sockpuppeteer; and the challenge is to find other fake reviews which are also written by the same author but using different aliases in future. Hence, the problem is reduced to an author verification problem. Given a few instances (reviews) written by a (known) sockpuppet author a, the task is to build an Author Verifier, AV_a (classifier) that can determine whether another (future) review is also written by a or not. This problem is related to authorship attribution (AA)[108] where the goal is to identify the author of a given document from a closed set of authors. However, having short reviews

with diverse topics render traditional AA methods, that mostly rely on content features, not very effective. While there have been works in AA for short texts such as tweets in [57] and with limited training data [66], the case for sockpuppets is different because it involves deception. Further, in reality sockpuppet detection is an open set problem (i.e., it has an infinite number of classes or authors) which makes it very difficult if not impossible to have a very good representative sample of the negative set for an author. In that regard, our problem bears resemblance with authorship verification [53].

In this work we, first, find that under traditional attribution setting, the precision of a verifier AV_a degrades with the increase in the diversity and size of $\neg a$, where $\neg a$ refers to the negative set authors for a given verifier AV_a. This is detailed in section 3.1.2. This shows that the verifier struggles with higher false positive and cannot learn $\neg a$ well. It lays the ground for exploiting the unlabeled test set to improve the negative set in training. Next, we improve the performance by learning verification models in lower dimensions. Particularly, we employ a feature selection scheme, ΔKL Parse Tree Features (henceforth abbreviated as ΔKL-PTFs) that exploits the KL-Divergence of the stylistic language models (computed using PTFs) of a and $\neg a$. Lastly, we address the problem by taking advantage of transduction. The idea is to simply put a carefully selected subset of positive samples, reviews authored by a (referred to as a spy set) from the training set to the unlabeled test set (i.e., the test set without seeing the true labels) and extract the nearest and farthest neighbors of the members in the spy set. These extracted neighbors (i.e., samples in the unlabeled test set which are close and far from the samples in the spy set) are potentially positive and negative samples that can improve building the verifier AV_a. This process is referred to as *spy induction*. The basic rationale is that since all samples retain their identity, a good distance metric should find hidden positive and negative samples in the unlabeled test set. The technique is particularly effective for situations where training data is limited in size and diversity. Although both spy induction and traditional transduction [114] exploit the assumption of implicit clusters in the data [12], there is a major difference between these two schemes; spy induction focuses on sub-sampling the unlabeled test set for potential positive and negative examples to grow the training

set whereas traditional transduction uses the entire unlabeled test set to find the hyper-plane that splits training and test sets in the same manner [47]. Our results show that for the current task, spy induction significantly outperforms traditional transduction and other baselines across a variety of classifiers and even for cross domains.

3.1.1 Dataset

Gokhman et. al. report that crowd-sourcing is a reasonable method for soliciting ground truths for deceptive content [31]. Crowd-sourcing has been successfully used for opinion spam generation in various previous works [83, 61, 60, 7]. In this work, our focus is to garner ground truth samples of multiple fake reviews written by one physical author (sockpuppet). To our knowledge, there is no existing dataset available for opinion spam sockpuppets. Hence, we used Amazon Mechanical Turk.

Participating Turkers were led to a website for this experiment where responses were captured. To model a realistic scenario such as singleton opinion spamming [117], Turkers were asked to act as a sockpuppet having access to several user-ids and each user-id was to be used exactly once to write a review as if written by that alias. The core task required writing 6 positive and 6 negative deceptive reviews, each had more than 200 words, on an entity (i.e., 12 reviews per entity). Each entity belonged to one of the three domains: hotel, restaurant and product. We selected 6 entities across each domain for this task. Each Turker had to complete the core task for two entities each per domain (i.e., 24 reviews per domain). The entities and domains were spread out evenly across 17 authors (Turkers). It took us over a month to collect all samples and the mean writing time per review was about 9 minutes.

To ensure original content, copy and paste was disabled in the logging website. We also followed important rubrics in [83] (e.g., restricted to US Turkers, maintaining an approval rating of at least 90%) and Turkers were briefed with the domain of deception with example fake reviews (from Yelp). All responses were evaluated manually and those not meeting the requirements (e.g., overly short, incorrect target entity, unintelligible, etc.) were discarded resulting in an average of 23 reviews per

Turker per domain. The data and code of this work is available at this link [2] and will be released to serve as a resource for furthering research on opinion spam and sockpuppet detection.

Throughout this work, for single domain experiments, we focus on the hotel domain which had the same trends to that of product and restaurant domains. However, we report results on all domains for cross domain analysis.

3.1.2 Hardness Analysis

This section aims to understand the hardness of sockpuppet verification via two schemes.

(I) **Employing Attribution**

An ideal verifier (classifier) for an author a requires a representative sample of $\neg a$. We can approximate this by assuming a pseudo author representing $\neg a$ and populating it by randomly selecting reviews of all authors except a. Under the AA paradigm, this is reduced to binary classification. We build author verifiers for each author $a_i \in A = \{a_1, ..., a_{17}\}$. As in the AA paradigm, we use an in-training setting, i.e., negative samples ($\neg a$) in both training and test sets are authored by the same closed set of 16 authors although the test and training sets are disjoint. Given our task, since there are not many documents per author to learn from, the effect of author diversity on problem hardness becomes relevant. Hence, we analyze the effect of the diversity and size of the negative set. Let $\lambda \in \{25\%, 50\%, 75\%, 100\%\}$ be the fraction of total authors in $\neg a$ that are used in building the verifier AV_a. Here λ refers to author diversity under in-training setting. We will later explore the effect of diversity under out-of-training setting. For example, when $\lambda = 50\%$, we randomly choose 8 authors, 50% of total 16 authors, from $\neg a$ to define the negative set for AV_a. Note that since we have a total of 16 authors in $\neg a$ for each a and all λ values, the class distribution is imbalanced with the negative class $\neg a$ in majority. We keep the training set balanced throughout this work as recommended in [80] to avoid learning bias due to data skewness. We use 5-fold Cross Validation (5-fold CV) so, the training fold consists of 80% of the positive (a) and

[2] https://www.dropbox.com/sh/xybjmxffmype3u2/AAA95vdkDp6z5fnTHxqjxq5Ga?dl=0

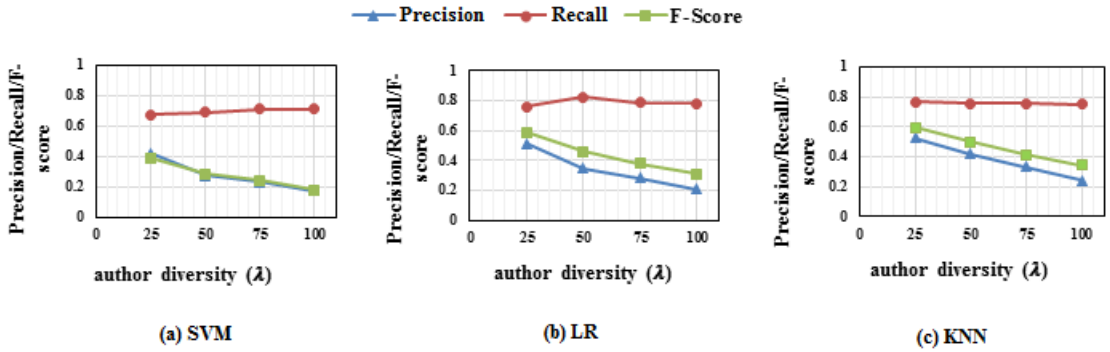
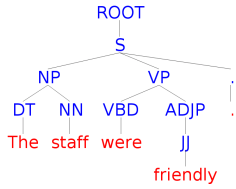


Figure 1: Effect of author diversity. Precision, recall and F-Score (y-axis) for different author diversity, $\lambda = 25\%, 50\%, 75\%, 100\%$ (x-axis) under in-training setting.

equal sized negative ($\neg a$) samples. But the test fold includes the remaining 20% of positive and remaining negative samples except those in training. Under this scheme, since $\neg a$ is the majority class in the test set, accuracy is not an effective metric. For each AV_a, we first compute the precision, recall and F-Score (on the positive class a) using 5-fold CV. Next, we average the results across all authors using their individual verifiers (Figure 1). This scheme yields us a robust measure of performance of sockpuppet verification across all authors and is used throughout this work. We report results of Support Vector Mechine (SVM), Logistic Regression (LR) and k-Nearest Neighbor (kNN) classifiers (using the libraries LIBSVM [11] for SVM with RBF kernel, LIBLINEAR[22] for LR with L2 regularization and WEKA for kNN with k=3 whose parameters were learned via CV)[3]. The feature space consists of lexical units (word unigram) and Parse Tree Features (PTF) extracted using the Stanford parser [51] with normalized term frequency for feature value assignment. Unless otherwise stated we use this feature set as well as the classifires setting for all experiments in this work. We followed some rules from [23] in computing PTFs. The rules are generated by traversing a parse tree in three ways i) a parent node to the combination of all its non-leaf nodes, ii) an internal node to its grandparent, iii) a parent to its internal child. We also add all interior nodes to the feature space (Table 1). From Figure 1, we note:

[3]http://www.cs.waikato.ac.nz/ml/weka/

Table 1: Four types of Parse Tree Feature (PTF)

Parse tree for: "The staff were friendly."	
	PTF(I) S → NP VP
	PTF(II) JJ ^ ADJP → VP
	PTF (III) S→ NP
	Interior nodes DT, NP

- With increase in diversity of negative samples, λ of $\neg a$, the test set size and variety also increase and we find significant drops in precision across all classifiers. This shows a significant rise in false positives. In other words, as the approximated negative set approaches the universal negative set ($\widetilde{\neg a} \rightarrow \neg a$ with increase in diversity of $\neg a$), learning $\neg a$ becomes harder.
- Recall, however, does not experience major changes with increase in the diversity of negative set as it is concerned with retrieving the positive class (a).
- F-Score being the harmonic mean of precision and recall, aligns with the precision performance order. We also note that F-Score in SVM and LR behave similarly followed by kNN.

Thus, sockpuppet verification is non-trivial and the hardness increases with the increase in $\neg a$ diversity.

(II) **Employing Accuracy and F1 on Balanced Class Distribution**

Under binary text classification and balanced class distribution, if accuracy or F1 are high, it shows that the two classes are well separated. This scheme was used in [53] for authorship verification. In our case, we adapt the method as follows. We consider two kinds of balanced data scenarios for a verifier for author a, AV_a: S_1 and S_2. Under S_1, we have the positive class P that consists of half of all reviews authored by a R_a, i.e., $P = \{r_i \in R_a; |P| = 1/2|R_a|\}$. The negative class N_{S_1} comprises of the other half, $N_{S_1} = \{r_i \in R_a - P; |N_{S_1}| = |P|\}$ and $S_1 = P \cup N_{S_1}$. Under S_2, we keep P intact but use a random sampling of $\neg a$ for its negative

Table 2: Classification results; P: Precision, R: Recall, Acc: Accuracy, F1: F-Score under two balanced data scenarios S_1 and S_2 for different classifiers.

	S_1				S_2			
Model	P	R	Acc	F1	P	R	Acc	F1
SVM	47.1	48.4	49.0	45.6	62.5	66.5	61.8	61.1
LR	47.4	46.4	49.5	44.6	63.5	67.4	61.7	62.1
kNN	41.9	57.4	49.8	44.9	51.0	68.9	56.1	53.8

class, $N_{S_2} = \{r_i \in R_{\neg a}; |N_{S_2}| = |P|\}$ yielding us $S_2 = P \cup N_{S_2}$. Essentially, with this scheme, we wish to understand the effect of negative training set when varied from false negative (N_{S_1}) to approximated true negative (N_{S_2}). Using lexical and parse tree features and 5-fold CV we report performance under each scenario S_1 and S_2 in Table 2. We note the following:

- The precision, recall, F1 and accuracy of all models under S_2 is higher than S_1. While this is intuitive, it shows for deceptive sockpuppets, writings of an author (P) bear separation from other sockpuppeters (N_{S_2}).
- Sockpuppet verification is a difficult problem because under balanced binary classification (S_2), there is just 5-10% gain in accuracy than random (50% accuracy). Yet it does show the models are learning some linguistic knowledge that separate a and $\neg a$ and using writings of authors other than a is a reasonable approximation for universal $\neg a$.

3.1.3 Learning in Lower Dimensions

The previous experiment suggests that in the case of deceptive sockpuppets, only a small set of features differentiate a and $\neg a$. As explored in [23], there often exist discriminative author specific stylistic elements that can characterize an author. However, the gamut of all PTFs per author (greater than 2000 features in our data) may be overlapping across authors (e.g., due to native language styles). To mine those discriminative PTFs, we need a feature selection scheme. We build on the idea of linguistic KL-Divergence in [80] and model stylistic elements to capture *how* things

are said as opposed to *what* is said. The key idea is to construct the stylistic language model for author, a and its pseudo author $\neg a$. Let A and $\neg A$ denote the stylistic language models for author a and $\neg a$ comprising the positive and negative class of AV_a respectively, where $A(t)$ and $\neg A(t)$ denote the probability of the PTF, t in the reviews of a and $\neg a$. $KL(A||\neg A) = \sum_t (A(t) \log_2 (A(t)/\neg A(t)))$ provides a quantitative measure of stylistic difference between a and $\neg a$. Based on its definition, PTF t that appears in A with higher probability than in $\neg A$, contributes most to $KL(A||\neg A)$. Being asymmetric, it also follows that PTF t' that appears in $\neg A$ more than in A contributes most to $KL(\neg A||A)$. Clearly, both of these types of PTF are useful for building AV_a. They can be combined by computing the per feature, f, ΔKL^f as follows:

$$\begin{aligned} \Delta KL_t^f &= KL_t(A_t||\neg A_t) - KL_t(\neg A_t||A_t), \\ KL_t(A_t||\neg A_t) &= A(t) \log_2 (A(t)/\neg A(t)), \\ KL_t(\neg A_t||A_t) &= \neg A(t) \log_2 (\neg A(t)/A(t)) \end{aligned} \tag{1}$$

Discriminative features are found by simply selecting the top PTF t based on the descending order of $|\Delta KL_t^f|$ until $|\Delta KL_t^f| < 0.01$. This is a form of sub-sampling the original PTF space and lowers the feature dimensionality. Intuitively, as KL_t is proportional to the relative difference between the probability of PTF t in positive (a) and negative ($\neg a$) classes, the above selection scheme provides us those PTF t that contribute most to the linguistic divergence between stylistic language models of a and $\neg a$.

To evaluate the effect of learning in lower dimensions, we consider a more realistic "out-of-training" setting instead of the in-training setting as in previous experiments. Under out-of-training setting, the classifier cannot see the writings of those authors that it may encounter in the test set. In other words test and training sets of a verifier AV_a are completely disjoint with respect to $\neg a$ which is realistic and also more difficult than in-training setting. Further, we explore the effect of author diversity under out-of-training setting, δ for the negative set (not to be confused with λ as in section 3.1.2). For each experiment, the reviews from $\delta\%$ of all authors except the intended author, $\neg a$ participate in the training of a verifier AV_a while the rest $(100 - \delta\%)$ authors make the

negative test set. We also consider standard lexical units (word unigram) (L), L + PTF, and top $k = 20\%$ (tuned via CV) PTF selected using χ^2 metric (L + PTF χ^2) as baselines. We examine different values of $\delta \in \{25\%, 50\%, 75\%\}$ but not $\delta = 100\%$ as that leaves no test samples due to out-of-training setting. From Table 3, we note:

- For each feature space, as the $\neg a$ diversity (δ) increases, across each classifier, we find gains in precision with reasonably lesser drops in recall resulting in overall higher F1. This shows that with increase in diversity in training, the verifiers reduced false positives improving their confidence. Note that verification gets harder for smaller δ as the size and skewness of the test set increases. This trend is different from what we saw in Figure 1 with λ which referred to diversity under in-training setting.

- Average F1 based on three classifiers (column AVG, Table 3) improves for $\delta = 25\%, 50\%$ using L+PTF than L showing parse tree feature can capture style. However feature selection using χ^2 (L+PTF χ^2) is not doing well as for all δ values there is reduction in F1 for SVM and LR. L+ΔKL PTF feature selection performs best in AVG F1 across different classifiers. It recovers the loss of PTF χ^2 and also improves over the L+PTF space by about 2-3%.

Table 3: P: Precision, R: Recall, F1: F-Score for out-of-training with different values of δ for three classifiers. AVG reports the average F1 across three classifiers. Feature Set: L: Lexical unit (word unigram), PTF: Parse tree feature, PTF χ^2 : PTF selected by χ^2 , ΔKL PTF: PTF selected via ΔKL

					δ=**25%**					
	SVM			**LR**			**kNN**			**AVG**
Feature Set	P	R	F1	P	R	F1	P	R	F1	F1
L	23.6	82.0	34.3	23.1	74.7	30.8	19.4	84.6	25.8	30.3
L+PTF	25.6	73.4	35.2	22.9	82.5	33.4	24.8	66.7	24.5	31.0
L+PTFχ^2	21.7	73.5	30.8	14.8	53.5	21.3	22.6	75.3	25.9	26.0
L+ΔKL PTF	25.6	79.2	36.3	21.7	80.2	32.1	22.3	81.5	27.8	**32.1**

(a)

					δ =**50%**					
	SVM			**LR**			**kNN**			**AVG**
Feature Set	P	R	F1	P	R	F1	P	R	F1	F1
L	30.7	83.6	41.8	28.7	83.1	38.7	21.1	85.1	27.1	35.9
L+PTF	33.2	73.4	42.7	30.6	78.1	40.9	28.0	73.8	28.8	37.5
L+PTFχ^2	24.8	69.2	33.7	21.0	47.8	26.9	23.4	81.6	30.2	30.3
L+ΔKL PTF	33.7	75.9	42.8	31.1	79.4	41.9	26.9	79.5	30.3	**38.3**

(b)

					δ =**75%**					
	SVM			**LR**			**kNN**			**AVG**
Feature Set	P	R	F1	P	R	F1	P	R	F1	F1
L	47.1	77.7	55.1	44.4	80.4	52.7	28.7	83.5	37.8	48.5
L+PTF	51.4	72.6	56.0	43.7	78.8	53.0	28.1	64.8	31.7	46.9
L+PTFχ^2	42.4	71.2	49.5	33.9	49.6	36.4	35.6	79.8	40.0	42.0
L+ΔKL PTF	50.5	71.9	56.2	46.3	79.4	54.9	42.2	80.8	46.1	**52.4**

(c)

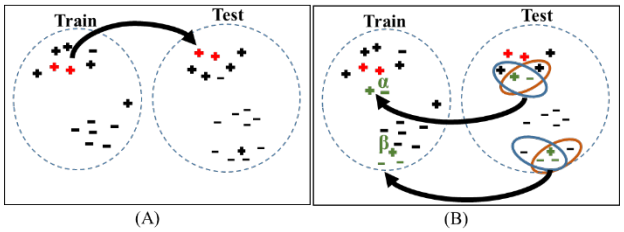


Figure 2: Spy induction : (A) spies (red plus signs) selected based on positive class centrality being put to the unlabeled test set. (B) common nearest and farthest neighbors (green plus and minus signs) across different spies' neighborhood shown by oval boundaries found in unlabeled test set being put back in the training set.

3.1.4 Spy Induction

We recall from section 3.1 that our problem suffers with limited training data per author as sock-puppets only use an alias few times. To improve verification, we need a way to learn from more instances. Also from section 3.1.2, we know that precision drops with increase in diversity of $\neg a$. This can be addressed by leveraging the unlabeled test set to improve the $\neg a$ set in training under transduction.

Figure 2 provides an overview of the scheme. For a given training set and a test set for AV_a, spy induction has three main steps. First is spy selection where some carefully selected positive samples are sent to the unlabeled test set. The second step is to find certain Nearest and Farthest Neighbors (abbreviated NN, FN henceforth) of the positive spy samples in the unlabeled test set. As the instances retain their original identity, a good distance metric should be able to retrieve potentially hidden positive (using common NN across different positive spies) and negative (using common FN across different positive spies) samples in the unlabeled test set. These newly retrieved samples from unlabeled test set are used to grow the training set. The previous step can have some label errors in NN and FN as they may not be true positive (a) and negative ($\neg a$) samples, which can be harmful in training. These are shown in Figure 2(B) by α_- and β_+ samples. To reduce such potential errors, a third step of label verification is employed where the labels of the newly retrieved samples from unlabeled test set are verified using agreement of classifiers on orthogonal feature spaces. with this step, we benefit from the extended training data without suffering from the

possible issue of error propagation. Lastly, the verifier undergoes improved training with additional samples and optimizes the F-Score on the training set.

(I) **Spy Selection**

This first step involves sending highly representative spies that can retrieve new samples to improve training. For a given verification problem, AV_a, let $D = D.Train \cup D.Test$ denotes the whole data. Although any positive instance in $D.Train$ can be a spy sample, only few of them might satisfy the representativeness constraint. Hence, we select the spies as those positive samples that have maximum similarity with other positive instances. In other words, the selection respects class based centrality and employs minimum overall pairwise distance (OPD) as its selection criterion:

$$OPD(s) = argmin_{s \in P}(\sum_{x \in P} d(s, x)) \tag{2}$$

where P is the positive class of training set, s denotes a potential spy sample and $d(\cdot)$ is distance function. Our spy set, $S = \{s\}$, consists of different spies that have the least pairwise distance to all other positive samples. We also consider different sizes of the spy set $|S| = n_S$ and experiment with different values of $n_S \in N_S = \{1, 3, 5, 7\}$. The method $SelectSpy(\cdot)$ (line 4, Algorithm 1 in Figure 3) implements this step.

(II) **New Instance Retrieval via Nearest and Farthest Neighbors**

After the selected spies are put into the unlabeled test set, the goal is to find potential positive and negative samples. Intuitively, one would expect that the closest data points to positive spy samples belong to the positive class while those that are farthest are likely negative samples. For each spy, $s \in S$, we consider n_Q nearest neighbors forming the likely positive set Q_s and n_R farthest neighbors forming the likely negative set R_s specific to s. Then, we find the common neighbors across multiple spies to get confidence on the likely positive or negative samples which yields us the final set of potentially Q positive and R

Algorithm 1: Spy induction

$SpyInduction(D, N_S, N_Q, N_R)$
1 : $P \leftarrow \{x \in D.Train, x.label > 0\}//positive\,class$
2 : $I \leftarrow \{(n_S, n_Q, n_R) | n_S \in N_S, n_Q \in N_Q, n_R \in N_R\}$
3 : $for\ each(i = (n_S, n_Q, n_R) \in I)$
4 : $\quad S \leftarrow SelectSpy(P, n_S)$
5 : $\quad Q \leftarrow ObtainNN(D.Test, S, n_Q)$
6 : $\quad R \leftarrow ObtainFN(D.Test, S, n_R)$
7 : $\quad (Q^v, R^v) \leftarrow CoLabelingVerification(Q, R, D.Train)$
8 : $\quad F1(i) \leftarrow CVImprovedTraining(D.Train, Q^v, R^v)$
9 : $endfor$
10 : $(n_S, n_Q, n_R)^* \leftarrow argmax\ _{i \in I}(F1(i))$
11 : $AV \leftarrow Classifier(D.Train, D.Test, (n_S, n_Q, n_R)^*)$

Figure 3: Spy induction algorithm

negative samples,

$$Q = \cap_{s \in S} Q_s; \quad R = \cap_{s \in S} R_s \tag{3}$$

This is implemented by the methods $ObtainNN(\cdot), ObtainFN(\cdot)$ (lines 5, 6, Algorithm 1). In most cases, we did not find the common neighbors Q, R to be empty, but if it is null, it implies no reliable samples were found. Further, like n_S, we try different values for $|Q_s| = n_Q; n_Q \in N_Q = \{1, 3\}$ and $|R_s| = n_R; n_R \in N_R = \{5, 10, 25, 40, 50, 60\}$. These values were set based on pilot experiments. The above scheme of new sample retrieval works with any distance metric. We consider two distance metrics on the feature space L+ΔKL PTF to compute all pairwise distances in the methods $SelectSpy(\cdot)$, $ObtainNN(\cdot)$ and $ObtainFN(\cdot)$ (lines 4-6, Algorithm 1): (1) Euclidean, (2) Distance metric learned from data. Specifically, we use the large margin method in [115] which learns a Mahalanobis distance metric $d_M(\cdot)$ that optimizes kNN classification in the training data using d_M. The goal is to learn $d_M(\cdot)$ such that the k-nearest neighbors (based on $d_M(\cdot)$) of each sample have the same class label as itself while different class samples are separated by a large margin.

(III) **Label Verification via Co-Labeling**

As it is not guaranteed that the distances between samples can capture the notion of authorship, the previous step can have errors, i.e., there may be some positive samples in R

and negative samples in Q. To solve this, we apply co-labeling [118] for label verification. In co-labeling, multiple views are considered for the data and classifiers are built on each view. Majority voting based on classifier agreement is used to predict labels of unlabeled instances. In our case, we consider $D.Train$ to train an SVM on five feature spaces (views): i) unigam, ii) unigram+bigram, iii) PTF, iv) POS , v) ΔKL PTF+unigram+bigram as five different label verification classifiers. Then, the labels of samples in Q and R are verified based on agreements of majority on classifier prediction. Samples having label discrepancies are discarded to yield the verified retrieved samples, (Q^v, R^v) (line 7, Algorithm 1). The rationale here is that it is less probable for majority of classifiers (each trained on a different view) to make the same mistake in predicting the label of a data point than a single classifier.

(IV) **Improved Training**

The retrieved and verified samples from the previous steps are put back into the training set. However, the key lies in estimating the right balance between the amount of spies sent, and the size of the neighborhood considered for retrieving potentially positive or negative samples, which are governed by the parameters n_S, n_Q, n_R. To find the optimal parameters, we try different values of the parameter triple, $i = (n_S, n_Q, n_R) \in I$ (lines 2, 3 Algorithm 1) and record the F-Score of 5-fold CV on $D.Train \cup Q^v \cup R^v$ as $F1(i)$ (line 8, Algorithm 1). This step is carried out by the method $CVImprovedTraining(.)$. Finally, the parameters that yield the highest $F1$ in training are chosen (line 10, Algorithm 1) to yield the output spy induced verifier (line 11, Algorithm 1).

3.1.5 Experimental Evaluation

This section evaluates the proposed spy induction method. We keep all experiment settings same as in section 3.1.3 (i.e., use out-of-training with varying author diversity δ). We fix our feature space to L+ΔKL PTF as it performed best (see Table 3). As mentioned earlier, we report average verification performance across all authors. Below we detail baselines, followed by results and sensitivity analysis.

(I) **Baselines and Systems**

We consider the following systems:

- **MBSP** runs the Memory-based shallow parsing approach [66] to authorship verification that is tailored for short text and limited training data.
- **Base** runs classification without spy induction and dovetails with Table 3 (last row) for each δ.
- **TSVM** uses the transductive learner of SVMLight [47] and aims to leverage the unlabeled (test) set by classifying a fraction of unlabeled samples to the positive class and optimizes the precision/recall breakeven point.
- **Spy (Eu.) & Spy (LM)** are spy induction systems without co-labeling but use Euclidean (Eu.) and learned distance metric (LM) to compute neighbors.
- **Spy (EuC) & Spy (LMC)** are extensions of previous models that consider label verification via co-labeling approach.

(II) **Results**

Table 4 reports the results. We note the following:

- Except for two cases (F1 of SVM and kNN for Spy(LM) with $\delta = 75\%$), almost all spy models are able to achieve significantly higher F1 than base (without spy induction) and TSVM for all classifiers SVM, LR, kNN and across all diversity values δ. MBSP performs similarly as Base showing memory based learning does not yield a significant advantage in sockpuppet verification. TSVM is not doing well on F1 but improves recall. One reason could be that due to class imbalance, TSVM has some bias in classifying unlabeled examples to positive class that improves recall but suffers in precision.
- The AVG F1 column shows that on average, across three classifiers spy induction yields at least 4% gain or more. The gains in AVG F1 are pronounced for $\delta = 25\%$ with gains upto 12% with spy (EuC). For $\delta = 75\%$, we find gains of about 10% in F1 with spy

Table 4: P: Precision, R: Recall, F1: F-Score results for spy induction under out-of-training with different values of δ; AVG: average F1 across three classifiers, Feature Set: L+ΔKL PTF. Gains in AVG F1 using spy (EuC) and (LMC) over baselines are significant at p<0.001 using a t-test

	SVM			**LR**			**kNN**			**AVG**
Model ($\delta = 25\%$)	P	R	F1	P	R	F1	P	R	F1	F1
MBSP	22.9	84.1	32.0	22.1	82.1	31.1	20.7	77.5	23.5	28.9
Base	25.6	79.2	36.3	21.7	80.2	32.1	22.3	81.5	27.8	32.1
TSVM	30.6	43.9	34.3	-	-	-	-	-	-	34.3
Spy(Eu.)	39.1	51.2	40.6	51.6	42.3	43.7	43.3	52.5	39.4	41.2
Spy(LM)	42.2	49.3	42.0	44.4	49.6	43.9	34.9	62.8	33.7	39.9
Spy(EuC)	42.0	57.8	42.7	51.2	61.3	52.5	41.5	57.9	38.4	**44.5**
Spy(LMC)	38.1	60.5	40.6	42.9	64.9	47.1	35.3	68.1	36.0	41.2

(A)

	SVM			**LR**			**kNN**			**AVG**
Model ($\delta = 50\%$)	P	R	F1	P	R	F1	P	R	F1	F1
MBSP	31.9	85.3	42.1	25.0	81.6	34.6	21.1	84.4	28.7	35.1
Base	33.7	75.9	42.8	31.1	79.4	41.9	26.9	79.5	30.3	38.3
TSVM	20.2	83.6	31.1	-	-	-	-	-	-	31.1
Spy(Eu.)	39.1	71.2	45.9	38.1	67.9	46.2	45.1	58.7	41.7	44.6
Spy(LM)	40.1	68.5	45.9	44.7	55.5	45.6	42.7	66.2	40.2	43.9
Spy(EuC)	62.3	52.0	52.3	62.5	64.6	61.0	46.6	62.3	43.2	**52.2**
Spy(LMC)	46.8	60.9	48.0	51.5	67.4	53.7	40.7	67.6	39.2	47.0

(B)

	SVM			**LR**			**kNN**			**AVG**
Model ($\delta = 75\%$)	P	R	F1	P	R	F1	P	R	F1	F1
MBSP	49.9	80.4	57.2	53.9	81.9	59.1	33.8	82.2	38.6	51.6
Base	50.5	71.9	56.2	46.3	79.4	54.9	42.2	80.8	46.1	52.4
TSVM	34.4	80.4	45.8	-	-	-	-	-	-	45.8
Spy(Eu.)	55.6	70.8	58.2	50.9	77.5	57.9	57.7	57.6	50.7	55.6
Spy(LM)	53.1	62.8	54.3	51.1	69.6	56.1	51.8	57.7	45.8	52.1
Spy(EuC)	71.9	59.1	62.4	68.9	75.6	70.2	63.6	59.0	54.7	**62.4**
Spy(LMC)	55.6	72.3	58.4	60.8	68.5	61.4	53.4	60.8	48.7	56.2

(C)

(EuC). Note that we employ out-of-training setting with varying author diversity (δ) so the test set is imbalanced (i.e., the random baseline is no longer 50%). Across all classifiers, the relative gains in F1 for spy methods over base reduce with increase in author diversity δ which is due (a) better $\neg a$ samples in training that raise the base result and (b) test set size and variety reduction limiting spy induction. Nonetheless, we note that for $\delta = 25\%$ (harder case of verification), spy induction does well across all classifiers.

- Anchoring on one distance metric (Eu./LM), we find that spy induction with co-labeling does markedly better than spy induction without co-labeling across all δ in AVG F1 across three classifiers. This shows label verification using co-labeling is helpful in filtering label noise and an essential component in spy induction.
- Between Euclidean and distance metric learned via large-margin (LM), Euclidean does better than LM in AVG F1 for both spy induction with and without co-labeling. However, using the LM metric yields higher recall than Euclidean in certain cases (underlined) which shows LM metric can yield gains in F1 beyond base with relatively lesser drops in recall which is again useful.

In summary, we can see that spy induction works in improving the F1 across different classifiers and author diversity and distance metrics. Overall, the scheme LR+Spy (EuC) does best across each δ (highlighted in gray) and is used for subsequent experiments to compare against Base.

(III) **Spy Parameter Sensitivity Analysis**

To analyze the sensitivity of the parameters, we plot the range of precision, recall and F1 values as spy induction learns the optimal values in training. We focus on the variation for $\delta = 25\%, 75\%$ capturing both extremes of diversity. Figure 4 shows the performance curves for different spy parameter triples (n_S, n_Q, n_R) sorted in the increasing order of F1. We find that for both $\delta = 25\%, 75\%$, the spy induction steadily improves precision with the increase in likely $\neg a$ samples (n_R). Although the recall drops more and has more fluctuations for the harder case of $\delta = 25\%$, it stabilizes early for $\delta = 75\%$ with much lesser drop in recall. This shows that the spy induction scheme is robust in optimizing F1 with only a few (5-7) spy samples (n_S) sent to unlabeled test set.

(IV) **Domain Adaptation**

We now test the effective of spy induction under domain transfer. As mentioned previously, we obtained reviews of Turkers for hotel, restaurant and product domains. Keeping all

Table 5: Cross domain results of LR + Spy (EuC). Gains in F1 using spy induction over base are significant at p<0.01 for all test domains and each δ using a t-test

	δ =25%		δ=50%		δ =75%	
Test Domain	Base	Spy	Base	Spy	Base	Spy
Hotel	30.3	36.4	40.0	47.0	50.3	52.6
Product	29.5	36.6	34.0	40.8	51.5	53.5
Restaurant	30.1	41.1	41.7	51.5	55.2	59.3

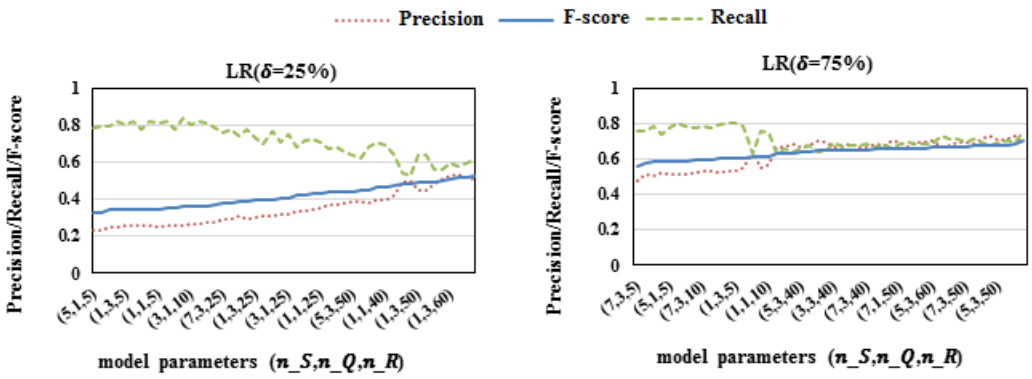


Figure 4: Spy parameter sensitivity. Variation of precision, recall and F-score (F1) across different parameter triples (n_S, n_Q, n_R)

other settings same as in Table 4, Table 5 reports results for cross domain performance by training the verifiers (AV_a) using two domains and testing on the third domain. We compare sockpupet verification using LR+Spy (EuC) vs. base (LR without spy induction). We report the F1 scores as the trends of precision and recall for cross domain were similar to the trends in Table 4. The F1 of base in cross domain (Table 5, Hotel row) is lower than corresponding LR results with base (Table 4) for all δ showing cross domain verification is harder. Nonetheless, spy induction is able to render statistically significant gains in F1 for all δ (see Table 5).

Table 6: Performance gains of Spy (EuC) in F1 over Base on Wikipedia sockpuppet dataset. Gains are significant ($p<0.01$) except for LR δ=25%, 50%, and 75%

	δ =25%		δ=50%		δ =75%	
Classifier	Base	Spy	Base	Spy	Base	Spy
SVM	50.7	**57.6**	59.6	**62.9**	68.0	**70.3**
LR	40.4	**42.4**	49.8	**51.1**	60.0	**61.5**
kNN	23.9	**29.5**	32.0	**37.1**	43.0	**51.1**

(V) **Performance on Wikipedia Sockpuppet (WikiSock) Dataset**

A corpus of Wikipedia sockpuppet authors is produced in [105]. It contains 305 authors with an average of 180 documents per author and 90 words per document which we use as another benchmark for evaluating our method. It is important to note that the base results reported in [105] are not directly comparable to this experiment (Table 6). This is because [105] used all 623 cases that were found as candidates but we focus on only 305 of them which were actually confirmed sockpuppets by Wikipedia administrators. Next, we perform experiments under realistic out-of-training setting and varying the author diversity (as in Table 4) which is different from [105]. This explains the rather lower F1 as reported in [105] for Base. We focus on F1 performance of spy (EuC) versus base (without spy) as the precision and recall trends were same as in Table 4. Compared to Table 4 base results, base does better for SVM and LR on WikiSock dataset that hints the data to be slightly easier. The relative gains of spy over base although are a bit lower than those in Table 4, spy induction consistently outperforms base.

3.1.6 Conclusion

This work performed an in-depth analysis of deceptive sockpuppet detection. We first showed that the problem is different from traditional authorship attribution or verification and gets more difficult with the increase in author diversity. Next, a feature selection scheme based on KL-Divergence of

stylistic language models was explored that yielded improvements in verification beyond baseline features. Finally, a transduction scheme, spy induction, was proposed to leverage the unlabeled test set. A comprehensive set of experiments showed that the proposed approach is robust across both (1) different classifiers, (2) cross domain knowledge transfer and significantly outperforms baselines. Further, this work produced a ground truth corpus of deceptive sockpuppets across three domains.

3.2 Authorship Verification

Authorship Verification (AV) is a branch of forensic authorship analysis. When given two text documents, we look to verify whether the two documents are written by the same author while no previous writing samples of their author/authors have been specified.

The majority of online services work on textual communications between users. Their overall reliability and performance can be impacted by someone who abuses the application and provides scripts while hiding their real identity and pretending to be someone else. To preserve the reliability of such services, the identity of the users should be monitored based on their provided scripts. The authorship verification techniques match the identity of the users with their writing styles. Indeed, authorship verification has an important impact on online document analysis such as plagiarism analysis, sockpuppet detection, blackmailing and email spoofing prevention, to name a few [37].

Traditionally, the studies on AV problem considered a closed and limited set of authors and a closed set of documents written by those authors. During the training step some of these documents (which were sometimes as long as a whole novel) were observed. Then, the problem was to identify whether the authors of a pair of the documents from the rest of the document set were identical [53, 67, 45]. This type of AV problem benefits from having access to the writing samples of *future authors* during the training step which is not always realistic. Actually, this structure is static and is not compatible with new future unseen authors. Recently, the structure of AV problem has changed and became more challenging. Based on the new structure, we are given some document pairs with their binary authorship status. The same-authorship status indicates that both are written by one author while the different-authorship status shows the pairs are written by two individual authors.

Based on this binary structure the goal is no longer to learn the writing style of each underlying authors individually (like in the traditional AV methods) but is to learn the difference or similarity of the writing styles of the two types of document pairs.

We address the problem of identifying the difference in documents from identical domains in two ways: 1- authorship *diversity* in *similar* contents by utilizing Amazon reviews from 300 distinct authors. 2- Scientific documents from the same area of research by different authors who have almost identical level of expertise in the field. It can also be considered as an application of plagiarism detection.

We analyze authorship verification on several datasets with binary structure. To our knowledge this amount of analysis has not been done in authorship verification on diverse types of datasets. The model is a Parallel Recurrent Neural Network (PRNN) that is inspired by the popular similarity measures in Statistical Machine Learning (ML). Being based on language models, it is mostly applicable for relatively larger datasets. PRNN compares the proximity of the language model of its two input sequences to investigate their authorship. We, also, propose the *summary vector* to adapt our problem to a common binary classification style to create strong baselines as there are limited studies in authorship verification according to the literature. Applying this adaptation, we are able to employ the recognized classifiers as well as similarity measures that are widely used in ML to build our baselines. Besides, the two pre-existing datasets, Amazon reviews and MPLA-400, are mapped to the binary structure to be used for our large scale AV problem.

Experimental results on evaluation datasets show that the model achieves stable and competitive performance compared to the baselines.

3.2.1 System Design

Let $P = (S, T)$ denotes a pair of documents, indicating S as the source and T as the target. Here, the task is to investigate whether S and T are written by the same author. We map this problem into a binary classification paradigm. Accordingly, if S and T are authored by the same person, P belongs to the positive class. Nevertheless, (S and T have different authors) P belongs

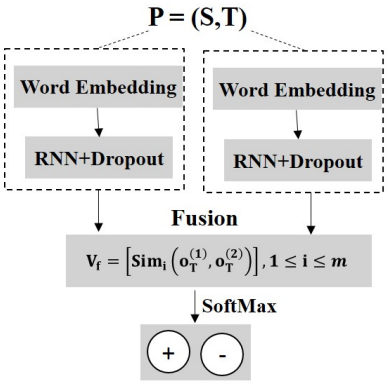


Figure 5: PRNN architecture. The network takes two inputs S, T in parallel and fuses them after passing word embedding and recurrent layers.

to the negative class. In the following section, we introduce the Parallel Recurrent Neural Network (PRNN) for large scale datasets.

3.2.2 Parallel Recurrent Neural Network (PRNN)

PRNN is designed to solve the AV problem for relatively large scale datasets. We model a pair of documents using a simple parallel recurrent architecture. The overall model is shown in Figure 5. In general, PRNN consists of three components: the parallel columns of identical layers, one shared fusion layer and a softmax layer as the output. We proceed to describe the network in the following paragraphs.

1. **Parallel columns of embedding and recurrent layers**

   Given two documents, the network takes each document as the input of one of the columns separately. In each column, the network embeds all words of the input document through an embedding matrix $E \in R^{d_E \times V_E}$ where V_E is the size of the vocabulary and d_E is the embedding dimension. Then, a fully-connected RNN where the output is to be fed back to its input takes the embedding matrix of the previous layer. The RNN layer of our model at

Table 7: Similarity functions; a, b: document vectors, n: number of features in a and b

Metric	Description	Metric	Description
Chi2 kernel	$\exp(-\gamma \sum_i [\frac{(a_i-b_i)^2}{(a_i+b_i)}])$	Cosine similarity	$\frac{ab^T}{\|\|a\|\|\|\|b\|\|}$
Euclidean	$(\sum_i (a_i - b_i)^2)^{0.5}$	Linear kernel	$a^T b$
RBF kernel	$\exp(-\gamma \|\|a - b\|\|^2)$	Mean of L1 norm	$\frac{\sum_i^n \|a_i-b_i\|}{n}$
Sigmoid kernel	$\tanh(\gamma a^T b + c_0)$		

time $t \in [0, \tau]$ includes:

$$
\begin{aligned}
h_t &= tanh(W^{hh} h_{t-1} + W^{hx} x_t + b) \\
o_t &= c + W^{ho} h_t
\end{aligned}
\tag{4}
$$

where x_t is the word embedding vector; b and c are the bias vectors; W^{hh}, W^{hx} and W^{ho} are the weight matrices. We only take o_τ, the output at the last time step τ, as the output of the recurrent layer. Finally, to avoid over-fitting problem we apply dropout regularization to the output of the recurrent layer. It helps the network to generalize the observed language models.

2. **Fusion layer**

Let o_τ^S and o_τ^T be the output of the final (RNN) layers of the two parallel columns after dropout. We add a *shared* fusion layer to fuse o_τ^S and o_τ^T by computing several popular similarity measures between them. The resulting fusion vector, V_f, is computed as: $V_f = [sim_1(o_\tau^S, o_\tau^T), ..., sim_M(o_\tau^S, o_\tau^T)]$, $V \in R^M$ where each $sim_{i,1 \leq i \leq M}$ function belongs to one of the M functions in Table 7. Finally, the output layer classifies the fusion vector using a softmax function.

3. **Dataset**

To evaluate PRNN we use all available authorship identification datasets released by PAN [4] (Table 8). Each PAN dataset consists of a training and test corpus and each corpus has

[4] http://pan.webis.de/data.html

Table 8: Authorship verification datasets statistics

Dataset	**Train**	**Test**
PAN2013	10	30
PAN2014E	200	200
PAN2014N	100	200
PAN2015	100	500
Dataset	**Positive**	**Negative**
Amazon	4500	4500
MPLA*	720	720

a various number of distinct problems. One problem is a pair of two documents: the first document of a problem composed of up to five writings (even as few as one) by a single person (implicitly disjoint For PAN2014 and PAN2015 and explicitly disjoint for PAN2013), and literally the second document includes one piece of writing. Two documents of a pair might be from significantly various genres and topics. The length of a document changes from a few hundred to a few thousand words. PAN2014 includes two datasets: Essays and Novels. The paired documents in PAN datasets are used for our experiments. So, for a problem $P = (S, T)$, S (source) is the first document and T (target) is the second document of a PAN problem. Besides, we evaluate PRNN on new schemas of MPLA-400 [5] and Amazon reviews. The schemas are defined similarly to the PAN-style explained above. MPLA-400 dataset contains 20 articles by each of the top-20 authors by citation in Machine Learning. We create its new schema, MPLA*, by selecting publications from MPLA-400 that are written by a single author and have no co-authors. To keep the distribution of authors and classes balanced in MPLA*, we select an equal number of single-authorship articles from all existing 20 authors and map it to the PAN-style (there are at most 9 single-author publications by each author). Here, the positive class consists of the pairs which are made up of all possible combinations of same-authorship articles ($20 \times \binom{9}{2} = 720$). And the same size negative class includes the pairs that are randomly selected from the set of all unique combinations of

[5] https://github.com/dainis-boumber/MLP-400-datasets

different-authorship articles.

We apply the similar method to Amazon review dataset to define its new PAN-style schema. We select 300 authors with at least 40 reviews to make the positive and negative candidate sets. Then, for each author, the positive candidate set is all possible and unique combinations of the author's reviews. To make the positive class we choose 4500 review pairs from this positive candidate set at random. On the other hand, the negative candidate set is made of all unique and possible combinations of review pairs having different authors. The negative class having equal size with the positive class is created by random selection from the negative candidate set.

3.2.3 Feature Sets and Experiment Settings

The plain text of each document is used as the input of PRNN. For the baselines, all documents of D^S and D^T are represented in vector space model under several feature sets with boolean feature value assignment separately. Seven feature sets are used: 1-unigram, 2-bigram, 3-trigram, 4-four-gram, 5-unigram Part Of Speech (POS), 6-bigram POS, 7-char-4gram [6]. We do not use the original training and test sets of the PAN datasets as the size of the training set is too small to be used for PRNN. To avoid overfitting we perform 5-fold Cross Validation (CV) for the PAN2015, Amazon and MPLA* where we have sufficient amount of examples in training folds. And for the PAN2013, PAN2014E and PAN2014N datasets that are relatively smaller we perform 10-fold CV to increase the size of the training folds. This setting is applied for PRNN as well as the baselines. We use Theano to implement PRNN. Default settings are used for classifier's parameters. The back-propagation is done using stochastic gradient descent with learning rate=0.001, batch size=1, and dropout rate=0.2. We use Glove pretrained vectors[7] as initial values for embedding vectors when there is a match. Otherwise, a random vector from a continuous uniform distribution over $[0, 1)$ is used.

[6] we use scikit-learn software for all linguistic features

[7] https://nlp.stanford.edu/projects/glove/

3.2.4 Comparison Methods

We connect several Machine Learning reliable classifiers widely used in the area with the seven similarity measures to set strong baselines (Table 7). Since each example in our underlying dataset comprises two documents, we need to adapt it to the structure of an ordinary classifier input by converting them to one single entity. A simple direct way is to concatenate their feature vectors. However, our experiments show it provides weak results mostly equal to the random label assignment. So, we define the *summary vector* as a single unit representative of each example/problem $P = (D^S, D^T)$ by utilizing several similarity measures. The summary vector comprises a class of several metrics each measures one aspect of the closeness of the two documents (D^S and D^T) of the pair for all underlying feature sets. For a pair of document feature vector (x, y) the summary vector is:

$$sum(x, y) = [sim_i^j(x, y)]_{1 \leq i \leq M, 1 \leq j \leq F} \tag{5}$$

where $sim_i^j(x, y)$ computes the ith similarity metric of M metrics in Table 7 under jth of $F = 7$ feature sets (Section 3.2.3) between x, y. Then, we use a classifier including **SVM**, Gaussian Naive Bayes (**GNB**), K-Nearest Neighbor (**KNN**), Logistic Regression (**LR**), Decision Tree (**DT**) and Multi-Layer Perception (**MLP**) to predict the class label.

3.2.5 Results and Analysis

The evaluation results are reported in Table 9. According to it, PRNN beats all baselines for all datasets except PAN2013 where it achieves the second highest accuracy. The best accuracy belongs to the Amazon dataset where we have the largest dataset. It can be inferred that when the scale of the underlying dataset is large enough, the network learns the difference between the two language models of its given inputs well. It should be noted that for the two PAN2013 and PAN2014E even after CV the network cannot converge and the validation loss increases after each epoch. To avoid it we increase the total number of document pairs by splitting each document into two smaller ones

Table 9: Classification accuracy using 5 and 10-fold CV across different AV datasets. The input for the baselines are empowered by the proposed similarity vector

Methods	MPLA*	Amazon	PAN2013	PAN2014E	PAN2014N	PAN2015
PRNN	**0.703**	**0.922**	0.72	**0.691**	**0.81**	**0.802**
SVM	0.621	0.818	0.525	0.659	0.673	0.628
NB	0.635	0.741	0.587	0.652	0.69	0.728
LR	0.671	0.839	0.581	0.676	0.707	0.675
KNN	0.64	0.831	**0.731**	0.656	0.75	0.757
DT	0.628	0.818	0.656	0.644	0.717	0.73
MLP	0.686	0.858	0.65	0.589	0.76	0.737

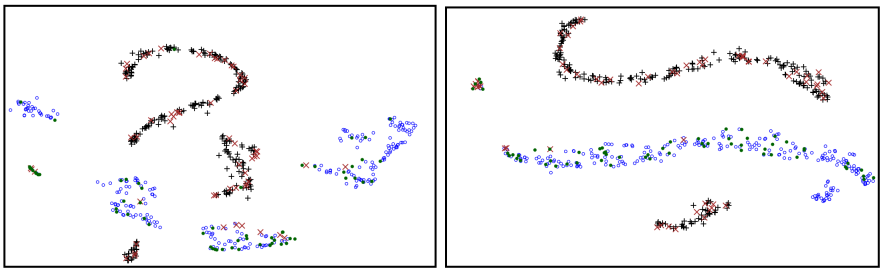


Figure 6: T-SNE plot of two folds of output of the fusion layer for PAN2015 in 5-fold CV. +: positive training data, ×: positive test data, ○: negative training data, •: negative test data

with an equal number of sentences and making new pairs. This technique decreases the validation loss during training. However, it still suffers from lack of labeled examples and causes weakest results compared to other larger datasets. To illustrate how PRNN discriminate writing styles we provide the t-SNE plot of the output of the fusion layer in a 5-fold CV classification for two folds of PAN2015 (Figure 6). According to Figure 6, both classes have almost similar distribution in the test and training data. But, in some rare parts, the positive and negative points are close. They are probably the portion of the data that mislead the classifier during the training step or will be misclassified in predictions.

Next, we improve our architecture for investigating any writing style change in documents.

3.3 Style Change Detection

Authorship identification methods usually assume that the underlying documents are authored by one single person. However, in a more realistic case, it is necessary to first check whether

the documents have one or multiple authors. Given one document, the problem of style change detection is to find if the writing style of the document has changed. In other words, we investigate whether it is written by multiple authors or one author. This is relatively a new problem in the area of mining writing style of text documents. Again, we need to study the writing style of a document by focusing on its linguistic aspects. Actually, the writing style expresses the selection of words and (grammatical) structure of a sentence.

According to the literature, there are two frequent NLP approaches for document representation. First, the Bag-of-Words (BoW) model that is independent of the word order of a sentence and expresses the *word selection* [56]. Second, the sequence models such as word embedding that are sensitive to the *order* of words of a sentence [69]. Although the English language is recognized to have a latent hierarchical, tree-based structure [15], none of the two approaches deploy the hierarchical structure of a sentence for its representation.

Similar to our sockpuppet detection task, we consider the latent structure of English language (Parse Tree) to represent a sentence. However, to preserve the order of the words in a sentence we no longer use the BoW model. We extract the ordered features of a parse tree that can be used by a parallel hierarchical attention network to find the determinative parts of a sentence and a document. Finally, the fusion layer of our verification task is used to compare a document with its reverse version to predict the class label. The results show that our model achieves promising results for the PAN 2018 dataset.

The model is described in details in Section 3.3.1. In Section 3.3.2 we provide the results and discussion.

3.3.1 Parallel Hierarchical Attention Network

Similar to our previous task in authorship verification [38], our model has a parallel structure with two columns of recurrent layers, one fusion, and one softmax layer. However, each column is no longer a simple RNN but is a hierarchical attention network with two levels of attention mechanism proposed in [121]. To be more specific, each column includes a Parse Tree Feature

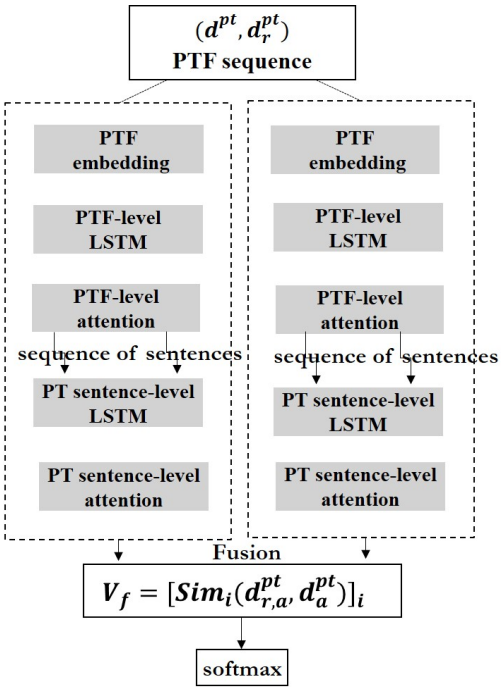


Figure 7: Parallel hierarchical attention network architecture

(PTF) embedding, a PTF-level LSTM, a PTF-level attention, a PT sentence-level LSTM, and a PT sentence-level attention layer. Here, the key difference is that the LSTM input is not the conventional character/word sequence but is the sequence of Parse Tree Features (PTFs) extracted from the tree-based structure of a sentence. The model architecture is shown in Figure 7. Each part will be described in the following sections.

1. **PTF Embedding**

   We use Stanford PCFG parser[8] to retrieve the hierarchical structure of a sentence [51]. Figure 8 shows the underlying parse tree for the sentence *"Computers beat chess players in 1980s."*. To use this tree structure by the following LSTMs in our model we need to preserve the word sequence of the sentence. We define Parse Tree Feature (PTF) of each word in a sentence as a path starting from the root to the corresponding leaf (word) of its parse tree. The path is a set of all rules of the form $parent \rightarrow child_1 ... child_n$ from root to a leaf (word). Here,

[8] https://stanfordnlp.github.io/CoreNLP/

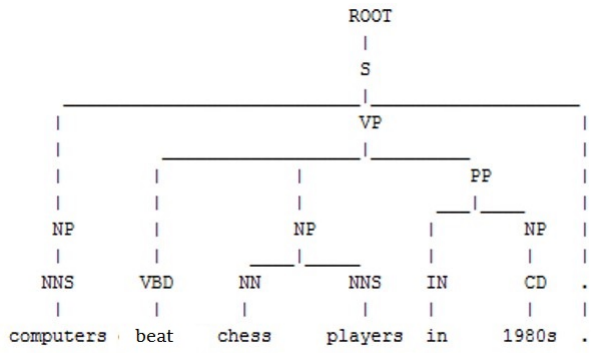


Figure 8: Parse tree for "Computers beat chess players in 1980s."

punctuation marks are considered as word unigrams. For example, the PTF of "computers" consists of three rules and is $[S \rightarrow NP\,VP\,., NP \rightarrow NNS, NNS \rightarrow computers]$ and PTF of "chess" with four rules is $[S \rightarrow NP\,VP\,., VP \rightarrow NP, NP \rightarrow NN, NN \rightarrow chess]$ (Figure 8). We ignore rule $ROOT \rightarrow S$ because it is a shared rule among all PTFs. Accordingly, the Parse Tree (PT) representation of a sentence is the set of PTF of all its word unigram. For the above example, s has seven PTFs where $PTF_{computer}$ is the first and $PTF_{.}$ is the last feature i.e., $s = [[S \rightarrow NP\,VP\,., NP \rightarrow NNS, NNS \rightarrow computers], ..., [S \rightarrow ., . \rightarrow .]]$

Let $d^{pt} = [s_i | i \in [0, n]]$ be Parse Tree (PT) representation of document d with n sentences where $s = [PTF_j | j \in [0, l_s]]$ is PT representation of sentence s with size l_s. As we mentioned earlier, we preserve the order of sentences and words according to their occurrence in a document. We define d_r^{pt} to be the reverse PT representation of d^{pt}. To make d_r^{pt}, first, we reverse the order of sentences of d^{pt} then the order of PTFs of each sentence. In other words, the last PTF of the last sentence of d^{pt} is the first PTF of d_r^{pt}. So, $d_r^{pt} = [s_{r,i} | i \in [n, 0]]$ where $s_r = [PTF_j | j \in [l_s, 0]]$ is the reverse of s. In our example, $s_r = [[S \rightarrow ., . \rightarrow .], ..., [S \rightarrow NP\,VP\,., NP \rightarrow NNS, NNS \rightarrow computers]]$.

2. **LSTMs and Attention Mechanism**

Here, the PT document representation and its reverse (d^{pt}, d_r^{pt}) are the inputs of our model,

each for one of the two columns. Later, we will explain why the PT reverse version of a document is one of the two inputs. Each column is a hierarchical attention network proposed in [121]. It has two layers of LSTM each followed by an attention layer. However, the input of the network is no longer word unigrams but is the PTFs. Besides, we use unidirectional LSTM instead of a bidirectional as we feed the reverse version of a document, almost similar to the backward pass of a bidirectional LSTM, to the second column of the network. The layers are described in the following:

- **Parse Tree Feature Embedding**: For a sentence $s = \{\text{PTF}_1, ..., \text{PTF}_{l_s}\}$ of l_s PTFs, we embed PTFs using PTF matrix embedding W_e that is initialized randomly and learned during the training phase. Here, $x_i = W_e\text{PTF}_i$ is the PTF embedding of the *ith* feature.

- **PTF-level LSTM and PTF-level Attention**: We choose LSTM in our model that is known to achieve promising results for long-term dependencies while its forget and update gates control the flow of information effectively. Here, h_i is the LSTM hidden state after *ith* feature of sentence s. Although we believe that PTFs express the writing style(s) of a given document, some are more determinative than others in predicting the class label. To highlight the importance of each PTF, we apply the attention mechanism proposed in [121] to the hidden state of PTF-level LSTM at step i. This mechanism computes a weight vector α_i using a Multi-Layer Perceptron and a softmax function. Here, W_{pt} and b are weight matrix and bias vector respectively, u_{pt} is a random context vector and will be adjusted during the training phase and s_a^{pt} is the sum of weighted hidden states of sentence s:

$$
\begin{aligned}
u_i &= \tanh(W_{pt}h_i + b), \\
\alpha_i &= \frac{\exp(u_i^\top u_{pt})}{\sum_i \exp(u_i^\top u_{pt})}, \\
s_a^{pt} &= \sum_i \alpha_i h_i
\end{aligned}
\tag{6}
$$

- **PT Sentence-level LSTM and PT Sentence-level Attention**

  The sequence of weighted sentences (s_a^{pt}) are the inputs of the PT sentence-level LSTM. Again, we would like to find which part of a document is more important for classification. In other words, we need to know in which sentence the style of the document is changed significantly. So, the PT sentence-level attention layer is applied to the hidden states of the PT sentence-level LSTM (h_j^s). Similarly, W_s and $b^{'}$ are weight matrix and bias vector respectively, u_s is a random context vector and will be adjusted during the training phase, β_j is the weight vector and d_a^{pt} is the final weighted document vector:

$$
\begin{aligned}
u_j^s &= \tanh(W_s h_j^s + b^{'}), \\
\beta_j &= \frac{\exp({u_j^s}^\top u_s)}{\sum_j \exp({u_j^s}^\top u_s)}, \\
d_a^{pt} &= \sum_j \beta_j h_j^s
\end{aligned}
\tag{7}
$$

  The reverse version of a document passes the same process through the second column of layers simultaneously. At this step, we have two weighted document vectors, the original (d_a^{pt}) and its reverse version ($d_{a,r}^{pt}$), from two parallel columns of layers. Next, we explain how to use the two document vectors for classification.

3. **Fusion and Output**

The last and important step is to investigate the style change in a document. In this problem, the number of authors is unknown to us and it is an open set problem with respect to the authors. Indeed, learning writing styles and observing a change may not be applicable solely. So, we need a mechanism independent of the number of authors/writing styles. Here, learning the *difference* between the two versions of a document is the key to find the existence of a style change. To do so, we use the fusion layer of our previous work where several similarity functions compute the similarity/difference between a pair of documents in a fully connected neural network layer [38]. The similarity functions are listed in Table 7. The weighted

document vector and its reverse version $d_a^{pt}, d_{r,a}^{pt}$ are compared in the fusion layer to learn the existence of a style change in documents by different authors and with various writing styles:

$$V_f = [sim_i(d_a^{pt}, d_{r,a}^{pt})]_i \tag{8}$$

where V_f is the similarity vector and sim_i belongs to one of the functions in Table 7. For documents with no style change, it compares the language model of one author between the forward and backward pass. However, for documents by multiple authors, there are two possible cases. In the first case, the order of different writing styles differs in both versions of a document. For example, document d by three authors $d = [author_1, author_2, author_3]$ and its reverse $d_r = [author_3, author_2, author_1]$. In the second case, the order of writing styles is the same in both regular and the reverse version of a document. For example, $d = [author_1, author_2, author_1]$ and $d_r = [author_1, author_2, author_1]$. For both cases, the fusion layer compares the writing style of multiple authors and the model learns the transition from one writing style to another. However, in the first case, the PT representation of the two documents are much more different than the second case as the order of authors differs in both versions of the document. Finally, the similarity vector V_f is given to a softmax function for binary classification.

3.3.2 Results and Analysis

We participate in PAN 2018 style change detection task [109, 48]. The task contains two training and validation sets that are publicly available before the competition and one test set which is not visible to the participants and is used to evaluate the participating models including our parallel attention network. Table 10 provides the dataset statistics. In the training phase, we use the negative log-likelihood as our loss function and RMSprop with learning rate $= 1e - 03$ as the optimizer. We initialize the 100-dimensional PTF embedding from a uniform distribution over $[0, 1)$. The size of the hidden layer of the two LSTMs is 8 with the batch size $= 1$. We also apply

Table 10: PAN2018 dataset statistics

feature	details
source	user posts from various sites of the StackExchange
#of documents	train: 2980, val.: 1492, test: 1352
#of authors	1-3
#of topics	15 (bicycles, Christianity,gaming and etc)
#of tokens/document	300-1000

a dropout of 0.3 on the output of the fusion.

Table 11 and Figure 9 show the evaluation results. We refer to our model as PHAN (Parallel Hierarchical Attention Network). During the competition, we did not use the word phrases as the last component of one PTF. PHAN without word phrase stays at the second rank with 82.5% accuracy on test set and 83.78% on validation set. As there is not much difference (less than 1.4%) between the accuracy of the two sets it indicates that the model is generalized well. Later, we consider word phrases when we were creating the PTFs. It results in more distinct features and an accuracy increase of 4.5%. To see the effect of utilizing PTFs in style change detection and fusion layer we do some experiments on PAN 2018 dataset.

1. **PTF vs Word Unigram**

   To show that PTFs are effective elements to represent one's style of writing, we train our model using only word unigram features instead of PTFs. We keep all settings intact and take the advantage of Glove pretrained word vectors as the input of embedding layer. Here, the reverse of a document is created as before and contains the set of reverse sentences from the last to the first sentence of the document. Results show that the accuracy of the validation set is 71%, almost 13% less than PTFs.

Table 11: PAN2018 dataset results. The results of other participants are reported from [48]

Methods	Accuracy
Zlatkova et al.	**89.3**
PHAN with word tokens	87
PHAN without word tokens	82.5
Safin and Ogaltsov	80.3
Khan	64.3
Schaetti	62.1

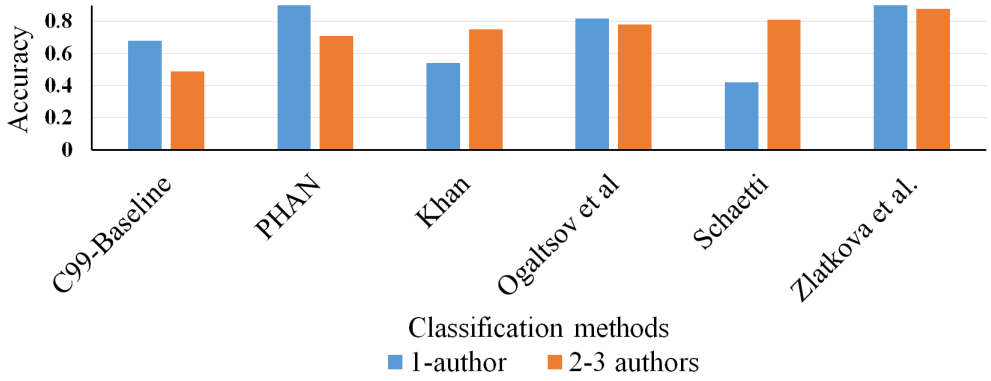


Figure 9: Accuracy of PAN 2018 models across positive (2-3-authors) and negative (1-author) classes [48]

2. **Fusion vs Concatenation**

The effect of the fusion layer that includes several well-known similarity metrics can be addressed with ablation. We replace the fusion layer with a fully connected layer that takes the concatenation of the two weighted document vectors produced from the two columns of attention networks. The accuracy on the validation dataset reduced by 10%.

The downside of this method is its expensive pre-processing phase that results in producing a huge PTF embedding dimensionality. The size of PTFs of the training set is around $1,300,000$ compared to $70,402$ word unigram features which is almost 19 times larger. However, this huge

dimensionality helps the attention mechanism to find and focus on discriminative features for class label prediction. [9] We believe the model can be improved if we learn the latent hierarchical structure of a sentence instead of using a pretrained tree structure of a sentence.

3.3.3 Conclusion

We propose a model to solve the new problem of style change detection in PAN 2018. We use parse tree features to deploy the hierarchical structure of a sentence and extract them such that the order of the corresponding words will be preserved to be used by a parallel hierarchical attention network. The results show that fusing the information from outputs of recurrent layers trained on only grammatical features achieves promising results.

[9] Producing PTFs using Stanford standalone parser made it slow and took around 10 hours in PAN evaluation process (test phase). It is much faster if one uses CoreNLP server available at https://stanfordnlp.github.io/CoreNLP/corenlp-server.html. However, we were not allowed to use any external resources during PAN evaluation phase.

4 STANCE DETECTION OF ARGUMENTATIVE OPINIONS

The problem of stance detection is to identify whether a given opinion supports an idea or contradicts it. It is relatively new in the area of opinion mining and is recently being explored by more researchers [5, 8, 14, 20, 35, 76]. Table 12 provides two arguments. The arguments answer a question while taking a stance of the two possible sides against a controversial issue. A stance that supports an issue is a *pro*, and the other side that is against it is a *con*.

In opinion mining identifying a stance of an opinion is a more challenging task than sentiment analysis [20] and naturally differs from it. Here, the problem is no longer finding the whole polarity of an opinion but is to identify its polarity against an issue. Recently, the argumentative opinions of controversial issues have attracted more people who want to take a stance after seeking enough information about the reason behind opinions from both sides. For example, one might wonder *"Should Marijuana Be a Medical Option?"*. This question might be found in many online debate forums and people who like to consume marijuana or the ones who hate it take a stance without bringing an acceptable justification. These types of opinions are usually short and express the stance directly (e.g. tweets). However, argumentative opinions are generally long, more complex, contain high-level ideas, and take a stance while bringing some reasons. Finding the stance of an argument is not straightforward compared to opinions with spontaneous language (e.g. tweets). See Table 12-pro as an example. We study the problem of stance detection in argumentative opinions of 46 different controversial issues. The arguments are collected and represented in a nonpartisan way which means that they are not biased specifically towards any party.

We make the following contributions in this work. First, we propose a new stance detection dataset from ProCon[10], a collection of critical controversial issues. Each entity of our ProCon dataset is a tuple of type (issue, question, context, argument) where an *issue* refers to the underlying domain, a *question* asks for an opinion, *context* brings a summary of proponent and opponent viewpoints about the *issue*, and an *argument* is a reason-based opinion for or against the *issue*.

[10]https://www.procon.org/

Table 12: Tow arguments, a pro and a con, for the issue ***medical marijuana***. The question is: ***"Should Marijuana Be a Medical Option?"***. Each example is a tuple of type (issue, question, context, argument).

Issue : Medical marijuana **Question** : Should Marijuana Be a Medical Option? **Context** : In 1970, the US Congress placed marijuana in Schedule I of the Controlled Substances ... Proponents of medical marijuana argue that it can be a safe and effective treatment for the symptoms of cancer, AIDS, multiple sclerosis, pain, glaucoma, epilepsy, and other conditions... Opponents of medical marijuana argue that it is too dangerous to use, lacks FDA-approval, and that various legal drugs make marijuana use unnecessary.
Argument (Pro): Ultimately, the issue is not about laws, science or politics, but sick patients. Making no distinction between individuals circumstances of use, the war on drugs has also become a war on suffering people. Legislators are not health care professionals and patients are not criminals, yet health and law become entwined in a needlessly cruel and sometimes deadly dance... I sincerely hope our work will illuminate the irrational injustice of medical marijuana prohibition.
Argument (Con): We can't really call marijuana medicine. It's not a legitimate medicine. The brain is not fully developed until we're about 25. That's just the way it is, and using any kind of mind-altering substance impacts that development. It needs to go through the FDA process...

Table 13 shows how people justify/condemn "legalization of abortion" while bringing some reasons.

We, also, propose a model that leverages the context of an issue to predict the stance of the given opinion. In ProCon dataset, the average number of opinions per issue per class is 24. This size of data may not be large enough for training a neural network. To compensate for this small size of data we build our model on top of the Universal pretrained Language Model, ULMFiT, [40] and fine tune it to our stance detection task. The pretrained language model is a "counterpart of ImageNet for NLP" and can be used in various tasks independent of document size and label as well as the number of in-domain documents [40].

The model is detailed in Section 4.2. We now discuss the related works. Then, we describe our dataset, the proposed model, and the experimental study.

Table 13: Two arguments about "legalization of abortion"

pro	con
The US Supreme Court has declared abortion to be a "fundamental right" guaranteed by the US Constitution. ... decision stated that the Constitution gives "a guarantee of certain areas or zones of privacy," and that "This right of privacy... is broad enough to encompass a woman's decision whether or not to terminate her pregnancy."	Unborn babies are considered human beings by the US government. The federal Unborn Victims of Violence Act, which was enacted "to protect unborn children from assault and murder", states that under federal law, anybody intentionally killing or attempting to kill an unborn child should be punished...for intentionally killing or attempting to kill a human being.

Table 14: ProCon dataset statistics; Docs refer to argumentative opinions (arguments)

	train		dev		test		train	
	docs	docs/issue	docs	docs/issue	docs	docs/issue	words/arg	words/cntx
size	1517	33	178	4	530	12	166 ± 65	177 ± 34

4.1 Dataset

We collect the information of 46 controversial issues from ProCon, a top-rated nonprofit organization that provides professionally-researched pros and cons to create our dataset (Table 14) [11]. We define each instance as a tuple of type I=(issue, context, question, argument) where the *issue* is a general topic, the *context* introduces the issue and brings a summary of proponent and opponent opinions and an *argument* is a reason-based opinion taking a stance on or against the given *issue* (Table 12). The issues cover various topics from health and medicine, education, politics, science and technology to entertainment and sports. We will use the words *target* and *issue* interchangeably in this work as *target* convey same meaning in other research. *Argument* supports a position with powerful and compelling statements. The dataset is divided into 1,517 train, 178 dev, and 530 test samples (Table 14).

[11] For more details visit https://www.procon.org/faqs.php

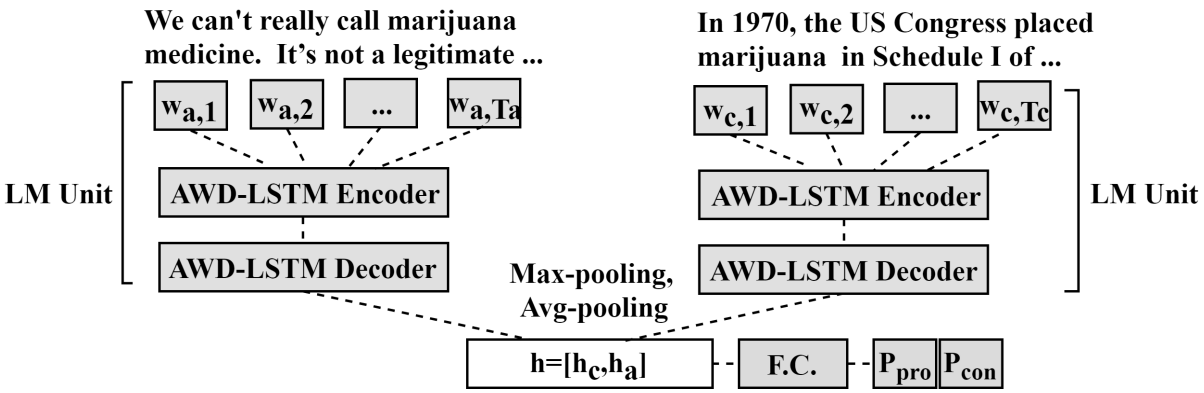


Figure 10: The model; $w_{a,i}, w_{c,i}$ ith word of argument and context sequence; F.C.: Fully Connected layer; p_{pro}, p_{con} : class probabilities; dark boxes use pretrained weights.

4.2 Model

Inspired by ULMFiT [40], we propose a model to handle both diverse and small training data per issue (Figure 10). Our model has three units: a) parallel Language Model (LM) units to learn an argument and the context of its underlying issue. b) one fusion unit that summarizes all elements of the data and c) the classification unit that predicts the stance. We describe them below.

4.2.1 Parallel LM Units

We let the model jointly learn an argument with its corresponding context using two LM units. A context usually covers a few sentences introducing the issue and two summaries of proponent and opponent arguments (Table12-context). We hypothesize that pro-arguments and con-arguments are related to disjoint parts of the context because of the intrinsic contradiction of pro- and con-arguments. Let $P = ([w_{1,a}, ..., w_{T_a,a}], [w_{1,c}, ..., w_{T_c,c}])$ be input pair where $w_{i,a}$ and $w_{i,c}$ are the ith word of an argument and its context sequence respectively and T_a, T_c are the last time steps. Each LM unit is a three-layer neural network (Figure 10). First, words are represented as vectors of size $d_e = 400$ using the embedding matrix W_e. The matrix is the result of pretraining the Language Model on Wikitext data with more than $103M$ words [68]. Then, a weight-dropped LSTM (AWD-LSTM) encodes word embedding to a higher dimension $(1,150)$, and another AWD-LSTM decodes the hidden representation of words into the embedding dimension and predicts the next word of the

sequence. AWD-LSTM applies recurrent regularization on the hidden-to-hidden weight matrices to prevent over-fitting across its connections. It adds Activation Regularization (AR) and Temporal Activation Regularization (TAR) to the loss function [68]. Later we provides more details of the two regularization techniques. The argument LM unit is the following:

$$\begin{aligned} x_{i,a} &= W_e w_{i,a}, \\ z_{i,a} &= \text{lstm}_{enc,a}(x_{i,a}), i \in [1, T_a], \\ h_{i,a} &= \text{lstm}_{dec,a}(z_{i,a}), i \in [1, T_a] \end{aligned} \tag{9}$$

where $z_{i,a}, h_{i,a}$ are the hidden state of LSTM encoder and decoder respectively. Similarly, $h_{i,c}$ is the output of context LM unit.

4.2.2 Pooling and Classification

The fusion layer leverages the information of both LM outputs. Most information of an argument is hidden in the last hidden state of the LSTM decoder of the LM unit. However, important information might be hidden anywhere in a *long* document. We use max-pooling and average-pooling of both inputs (argument, context) along with the last hidden state of LSTM decoder for fusion.

$$\begin{aligned} h_a &= [h_{T_a,a}, \text{max-pool}(h_{T_a,a}), \text{avg-pool}(h_{T_a,a})], \\ h_c &= [h_{T_c,c}, \text{max-pool}(h_{T_c,c}), \text{avg-pool}(h_{T_c,c})] \end{aligned} \tag{10}$$

where $h_{T_a,a}$,$h_{T_c,c}$ are the hidden state of LSTM decoder of argument and context LM units at time T_a, T_c and $[,]$ is concatenation. Finally, the pooled information, $h = [h_a, h_c]$, builds the fusion layer and connects an argument with any significant parts of the context. We feed h through a fully connected layer with $d_r = 50$ hidden neurons activated with a rectifier. The second fully-connected layer but with the linear activation gives us 2d vectors to be used by a softmax function for classification. We apply batch-normalization and dropout to both fully-connected layers to

avoid over-fitting. As we mentioned earlier, AWD-LSTM adds TAR (l_{tar}) and AR (l_{ar}) to the final loss. AR is an L2-regularization that controls the norm of the weights to reduce over-fitting. And TAR acts as L2 decay and is used on individual activations. It considers the difference of the outputs of the LSTM decoder at consecutive time steps:

$$\begin{aligned} l_{ar} &= \alpha * |||[h_{T_a,a}, h_{T_c,c}]|||_2, \\ l_{tar} &= \beta * |||[h^{'}_{T_a,a}, h^{'}_{T_c,c}] - [h^{'}_{T_a-1,a}, h^{'}_{T_c-1,c}]|||_2, \\ L &= -\sum_{d} \log h_{s,j} + l_{ar} + l_{tar} \end{aligned} \tag{11}$$

where j is the label of the document and $\alpha = 2$, $\beta = 1$ are the scaling coefficients. $h^{'}_{T_a,a}, h^{'}_{T_c,c}$ are the last hidden states of the two LSTM decoders without dropout.

4.3 Experimental Evaluation

We compare our model with state-of-the-art methods in stance detection. The methods are as follows:

- BoW-s : is a Bag of Words model that gains the best performance in TaskA of SemEval2016 [76] with SVM classifier. The features are boolean representation (0/1) of word uni-, bi- and tri-grams as well as character 2, 3, 4 and 5-grams. The presence/absence of any manually selected keywords of the underlying issue is also added to the feature vector. For example, for the issue of 'Hillary Clinton' the presence of *Hillary* or *Clinton* sets this feature to true. We manually select at least three keywords per issue in Procon dataset. Unlike [76], we do not build an individual classifier for each issue separately. We create one general classifier trained on the whole dataset. We examine the BoW-s feature vectors with SVM, Gaussian Naive Bayes (GNB), Logistic Regression (LR), and Random Forest (RF).[12]

- Independent Encoding (IE) : is one of the baselines reported in [5]. It learns the representation of a document and its target independently using two parellel LSTMs. Then, the last hidden

[12] we use scikit-learn with default settings

states of the two LSTMs are concatenated and projected with the *tanh* function. Finally, a softmax predicts the class distribution over the non-linear projection.

- ULMFiT: is the backbone of our model [40]. We keep all settings intact and train the model by applying the discriminative fine-tuning technique.

- Bidirectional Conditional Encoding (BiCoEn): outperforms the existing methods of SemEval 2016-TaskB. In TaskB the goal is to predict the stance of a tweet over one single unseen target, 'Donald Trump'[5]. The model takes a *tweet* and its underlying *target* and initializes the state of the bidirectional LSTM of tweets with the last hidden state of the forward and backward encoding of the target. In this way the model builds target-dependent representations of a tweet while both the left and right sides of a word are considered. This model takes a document (tweet) and its target (e.g. 'Climate Change is a Real Concern' or 'Atheism') as input for training. To make the comparison more reliable we examine BiConEn for both types of input: (*argument*, *context*) and (*argument*, *issue*). Here, *issue* is the *target* as in [5].[13]

4.4 Results and Analysis

We apply discriminative fine-tuning for ULMFiT and our model. We execute the evaluation 5 times and report the average results for all methods. Table 15 provides the experimental results. The largest values are highlighted in bold and the second largest are underlined. According to the table, both accuracy and Macro-F1 of all baselines do not exceed 65%, showing that the presence of diverse issues makes the problem hard to solve. It is expected that the Neural Network (NN) baselines give weak results compared to BoW for ProCon data. With $1,517$ training samples and 46 different issues, the average number of arguments per issue is 33 which is not enough for fitting NN models unless we provide some external knowledge for them such as what we do for our model (Pre-trained Language Model). It notes that stance detection is not a pure binary classification problem, because detecting the underlying issue is required for identifying the polarity of opinion

[13]We use their code shared on https://github.com/sheffieldnlp/stance-conditional

Table 15: ProCon19 dataset results; arg: argument, cntx: context, P:Precision, R:Recall, M-F1: Macro-F1

Method	Input	P	R	F1	P	R	F1	**M-F1**	**Acc**
			Pro			Con			
BoW-s+SVM	(arg)	0.61	0.63	0.62	0.62	0.61	0.61	0.62	0.62
BoW-s+RF	(arg)	0.64	0.67	0.65	0.66	0.62	0.64	0.65	0.65
BoW-s+LR	(arg)	0.61	0.63	0.62	0.62	0.6	0.61	0.61	0.61
BoW-s+GNB	(arg)	0.58	0.65	0.62	0.61	0.54	0.57	0.59	0.59
ULMFiT	(arg, cntx)	65.8	61.2	63.4	64	**68.5**	**66.2**	64.8	64.9
IE	(arg, issue)	55.4	60.5	57.5	56.7	51.1	53.2	55.4	56.7
IE	(arg, cntx)	56.9	52.7	54.5	56.1	60.1	57.8	56.2	57.3
BiCoEn	(arg, issue)	56.5	57.7	57	57.2	55.9	56.4	56.7	56.9
BiCoEn	(arg, cntx)	55.9	57.6	56.4	56.7	54.6	55.2	55.8	56.6
Our model	(arg, cntx)	**65.9**	**82.6**	**73.3**	**77**	57.7	65.9	**69.6**	**70.1**

against it. Aside from the above notes, BiCoEn is designed for detecting the stance of tweets for one unseen single target (issue), however, in ProCon the size of input argument is much longer than a tweet (166 compared to 20 words) and belongs to a diverse number of issues. We set the maximum length of an input to be 20 words for IE and BiConEn, as recommended by the authors of [5]. However, we find that by increasing this threshold, accuracy decreases. The reason is that the sequence length of both LSTMs must be equal, because the initial weights of argument-LSTM are the output of issue-LSTM. When we increase the maximum length, issue-LSTM takes no new information but padding indices (the average length of argument sequence is much greater than average length of issue sequence, $166 \gg 3$).

Ultimately, our model achieves an accuracy increase of more than 5% compared to BoW+RF. It indicates that leveraging the context information along with LM Fine-Tuning helps the model identify the issue and the stance against it more accurately. We provide more analysis in the following sections.

Figure 11: Heatmap of max-pooling matrix of one argument. The underlying question: *"Is the Use of Standardized Tests Improving Education in America?"*. Darker colors show larger scores

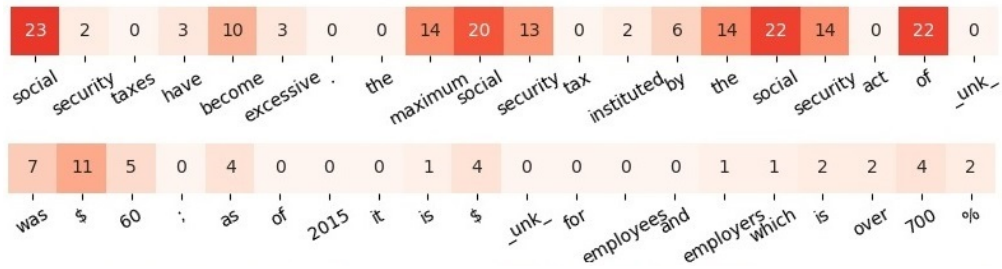


Figure 12: Heatmap of max-pooling matrix of the first half of an argument (the second half scores are mostly zero). The underlying question: *"Should Social Security Be Privatized?"*. Darker colors show larger scores.

4.4.1 Effect of Max-pooling

The fusion layer merges the information from previous layers for prediction. To understand what the model learns in this layer we plot the word scores in the max-pooling matrix of an argument. We define the score of word w at time t, to be the index frequency of the embedding vector of w in pooling operation. The larger the score, the more important that word is to the model, because more embedding dimensions of that word appear in the max-pooled matrix (same word in different time steps may have different scores). Figures 11 and 12 show the heatmaps of a short and the first half of a longer argument respectively that are correctly classified. We cannot provide more plots due to space constraints. However, we find that the words at the beginning of long documents are more informative (Figure 12). One reason is that the first sentence of long arguments is *usually* the topic sentence that conveys the stance. Moreover, for shorter arguments, the model finds the information across all parts of the argument almost evenly.

Table 16: Effect of LM fine-tuning for ProCon19 dataset

Method	LM-FT	P	R	F1	P	R	F1	**F1**	**Acc**
			Pro			Con			
Our model	no	53.2	68.6	59.9	56.3	40.2	46.9	53.4	54.3
Our model	yes	65.9	82.6	73.3	77	57.7	65.9	69.6	70.1

4.4.2 Effect of Pre-trained LM

To assess the impact of pre-trained LM, we examine our model without utilizing the pre-trained LM. We do not fine-tune the LM units over the training data, too. The experiment are represented in Table 16. The dramatic drop in all metrics shows the effect of the ablated techniques. Pre-training helps generalization and prevents our model from overfitting the relatively small training data.

4.4.3 Sentiment Analysis

How does sentiment relate to stance? Are pro-opinions often positive while cons are negative? To answer this question and find the relation between the stance and sentiment we define the sentiment score s_d of document d as $s_d = \sum_{s \in d} s_v$ where s_v is the VADER sentiment score of sentence s [29]. We compare the average sentiment score of the 23 issues from training set arguments between two classes (Figure 13). According to the plot, in some cases such as *abortion* and *voting machines* the score of pro is positive while con has negative overall score, indicating that proponents and opponents have different sentiments in their arguments about the issue. For some other cases, such as *health care*, both classes have a positive sentiment score. We identify as a key reason the concept of *'the right to health'*, has a positive sentiment. That makes opponents use this concept and its synonyms frequently making their arguments statistically positive. For "churches" where the underlying question is "should churches remain tax-exempt?" con has a larger positive score than pro. We find that some supporters (pro class) bring negative justifications by predicting the unsatisfactory situation after withdrawing the tax-exempt for churches. This unsatisfactory situation is explained while having negative sentiment.

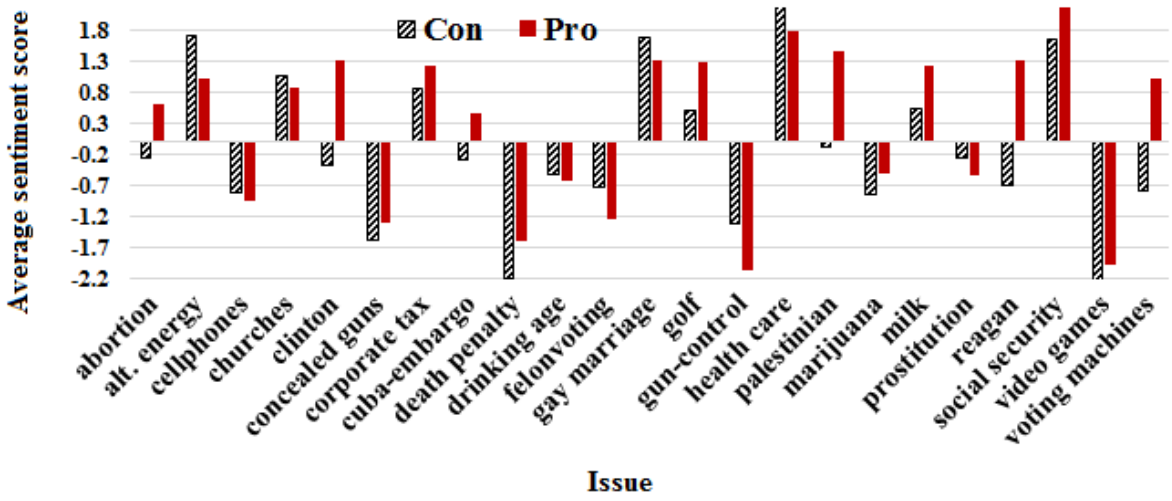


Figure 13: Average sentiment score per issue per class

4.5 A Transformer-Based Model for Stance Detection

Utilizing pre-trained models has been widely popular in machine translation and various text classification tasks. Prior efforts were hindered by lack of labelled data. With the growth of successful pre-trained models, a model fine-tuned on a small portion of data can compete with models trained on 10X more training data without pre-training [41]. Recently, transformer models trained on both directions of language simultaneously, such as BERT [19] and XLNet [120], overcome previous unidirectional language models (e.g. ULMFiT [41]) or models trained on two independent direction (ELMo) [87] significantly. So, we build our baselines based on BERT architecture in two different ways: single and pair of inputs. A question and its related opinion are concatenated for single inputs. However, for input pairs question and the opinion are being separated with the BERT separator tag [SEP]. This approach has been used for question-answering applications.

In this work, we leverage sentiment information with the BERT representation obtained from the last BERT-base layer as the input of a shallow recurrent neural network. The analysis in [36] shows that sentiment varies in pro and con-opinions among different issues. For some issues, both sides have equal average positive or negative sentiments, while positive sentiment is less present in

proponents' opinions than opponents' opinions for some other issues, such as tax. One of our goals is to analyze how sentiment of sentences can affect stance detection. We do not manually label the data, but use VADER [42], which is a rule-based sentiment tool for obtaining the sentiment of a sentence. For each sentence, its compound VADER score, ranging from -1 to $+1$, is translated into one of the three positive, neutral or negative sentiment labels. The sentiment of each token is the same as its corresponding sentence. Then, these labels are embedded using $W^s_{3\times d}$ and are concatenated with token embeddings to form our recurrent model input. $d = 50$ is the sentiment dimension.

$$
\begin{aligned}
x_t &= [h_t^{\text{BERT}}; e_t^{\text{snt}}], \\
z_t &= \text{RNN}(x_t), \text{RNN} \in [\text{LSTM}, \text{GRU}] \\
u &= [\text{avg-pool}(Z); \text{max-pool}(Z); z_T], \\
y &= \text{softmax}(Wu + b)
\end{aligned}
\tag{12}
$$

For an input sequence with T tokens, h_t^{BERT} is hidden state of the last BERT-base layer corresponds to the input token at time t, e_t^{snt} is sentiment embedding of the token, [;] concatenation operator, $Z = [z_i]_{i=1}^T$, and W, b are parameters of a fully connected layer. RNN is either Long Term Short Memory (LSTM) network or Gated Recurrent Units (GRU). Because important information towards the final stance of an opinion might be anywhere in a document, we pool the information from the recurrent hidden states of all input tokens using max and average-pooling. The last hidden state of the recurrent network (z_t) is used with pooled information for classification. Finally, a dense layer transforms them to class dimension. Figure 14 shows the model architecture.

We call our networks Sent-LSTM and Sent-GRU and report their experimental results in Section 4.5.3

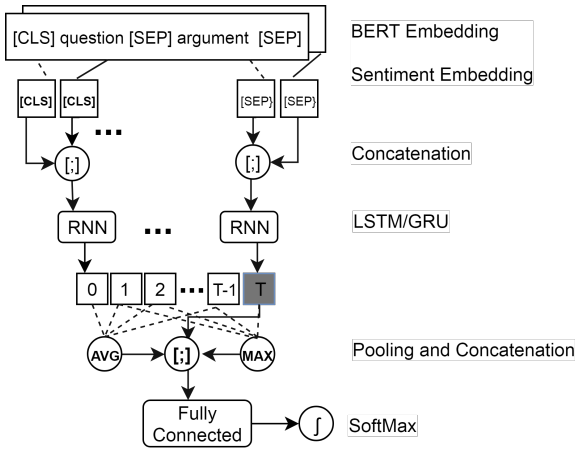


Figure 14: Stance detection architecture

4.5.1 Experiments

Similar to our prior effort in Procon19 [36], we collect data from procon.org. It gives us 419 different issues with 6000 examples. Some examples of questions from dataset are provided in Table 17. We use the following baselines:

- BERT [19] followed by a nonlinear transformation on a dense layer is used for downstream stance detection. Here, the whole network is fine-tuned and the all 12 BERT-base layers' weights will be updated in backpropagation. The information is pooled from the hidden state of the classification token ($h^{\text{BERT}}_{[\text{cls}]}$) of input sequence after passing a fully connected layer with non-linear activation (tanh). Then, a classifier layer shrinks the activations to a binary dimension.

$$x = \tanh(W^p h^{\text{BERT}}_{[\text{cls}]} + b^p),$$

$$y = W^c x + b^c$$

  where $W^c, W^p, b^p,$ and b^c are the layers' parameters.

- XML-CNN model consists of three convolution layers with kernel size= $(2, 4, 8)$. With a dynamic max-pooling layer, crucial information are extracted across the document. XML-CNN was able to beat most of its deep neural network baselines in six benchmark datasets

Table 17: Examples of ProCon20 dataset questions

HEALTH and MEDICINE 1- Should Euthanasia or Physician-Assisted Suicide Be Legal? 2- Is Vaping with E-Cigarettes Safe?
EDUCATION 1-Should parents or other adults be able to ban books from schools and libraries? 2- Should Public College Be Tuition-Free?
POLITICS 1- Should Recreational Marijuana Be Legal? 2- Should More Gun Control Laws Be Enacted?
SCIENCE and TECHNOLOGY 1- Is Cell Phone Radiation Safe? 2- Should Net Neutrality Be Restored?
ENTERTAINMENT and SPORTS 1- Are Social Networking Sites Good for Our Society? 2- Do Violent Video Games Contribute to Youth Violence?

[63]. We use both BERT representation as well as *Word2vec* Embedding for input.

- AWD-LSTM [68] is weight-dropped LSTM that deploys DropConnect on hidden-to-hidden weights as a form of recurrent regularization. *Word2vec* Embedding is used for input.

We define the corresponding hidden states of the *last BERT encoder layer* as BERT embedding/representation of input sequence for both single and pair of inputs mode. The BERT embedding are used as the input of Sent-LSTM and Sent-GRU models.

4.5.2 Training

We develop our code based on the Hedwig[14] implementation and train the models on 30 epochs with batch size=16. We apply early stopping technique to avoid over-fitting during training. Training is stopped after 5 consequent epochs of no improvement of highest balanced accuracy. Balanced accuracy computes the average of recall acquired on each class and is equal to accuracy for balanced

[14]https://github.com/castorini/hedwig

Table 18: Procon20 results; Pr.:Precision, Re.:Recall, Acc.:balanced accuracy

Model	Set	Pr.	Re.	F1	Acc.
pair of input					
BERT	dev	79.65	84.05	81.79	79.97
BERT	test	74.74	**81.57**	**78.0**	**76.85**
XML-CNN$_{\text{BERT}}$	dev	79.87	74.23	76.95	76.19
XML-CNN$_{\text{BERT}}$	test	74.12	72.43	73.27	73.40
Sent-LSTM	dev	81.88	74.85	78.21	77.67
Sent-LSTM	test	**77.76**	71.29	74.38	75.29
Sent-GRU	dev	79.75	79.75	79.75	78.33
Sent-GRU	test	74.85	80.10	77.38	76.44
unary input					
BERT	dev	78.18	79.14	78.66	77.01
BERT	test	76.84	76.84	76.84	76.68
XML-CNN$_{\text{BERT}}$	dev	76.60	77.30	76.95	75.21
XML-CNN$_{\text{BERT}}$	test	73.28	74.71	73.99	73.56
AWD-LSTM	dev	67.32	73.31	70.19	66.23
AWD-LSTM	test	60.43	64.27	62.29	60.85
XML-CNN$_{\text{Word2vec}}$	dev	68.32	67.48	67.90	65.78
XML-CNN$_{\text{Word2vec}}$	test	66.01	65.25	65.63	65.63
Sent-LSTM	dev	76.88	78.53	77.69	75.86
Sent-LSTM	test	70.76	76.18	73.37	72.17
Sent-GRU	dev	81.27	70.55	75.53	75.53
Sent-GRU	test	77.59	68.35	72.68	74.14

data. We inspect the test set on the model with the best balanced accuracy of development set and keep the settings for BERT the same as *BERT-base-uncased* model. BERTAdam optimizer with the learning rate of $2e-5$ recommended in [19] is used. We see dramatic drop in BERT performance with some other learning rates. Scikit-learn [84] library is employed for evaluation measures.

4.5.3 Results and Discussion

Experimental results are provided in Table 18. It was expected that fine-tuning BERT with pair of input gains highest performance, but, it shows that even with shallow concatenation of the question and argument (unary input) BERT can achieve reasonable results. Among the models with pair of input Sent-GRU, denoted with underline, shows the second highest performance and closest to

Table 19: Effect of sentiment for ProCon20 dataset

Model	F1	Acc.
Sent-LSTM	74.38	75.29
LSTM	74.15 (**0.23** ↓)	73.73 (**1.56** ↓)
Sent-GRU	77.38	76.44
GRU	73.41 **(3.97 ↓)**	74.96 (**1.48** ↓)

BERT in all measures (Sent-LSTM outperforms BERT in terms of precision). It indicates that the external knowledge gained from a massive corpora fine-tuned on 20X fewer parameters can compete with the original architecture. As the model is significantly smaller, it trains remarkably faster.

1. **Effect of Flow of Sentiment** As stated in Section 4, our recurrent models (Sent-LSTM, Sent-GRU) integrate sentiment of each sentence in token-level and learn the flow of sentiment across an argument. To see the effect of sentiment, we lift sentiment embeddings from the input of the models. So, LSTM and GRU models are simple Long Term Short Memory and Gated Recurrent Units network, respectively followed by pooling and classification layers:

$$
\begin{aligned}
x_t &= h_t^{\text{BERT}}, \\
z_t &= \text{RNN}(x_t), \text{RNN} \in [\text{LSTM}, \text{GRU}], \\
u &= [\text{avg-pool}(Z); \text{max-pool}(Z); z_T], \\
y &= \text{softmax}(Wu + b)
\end{aligned}
\tag{13}
$$

   For an input sequence with T tokens, h_t^{BERT} is the hidden state of the last BERT layer, corresponding to the input token at time t, $Z = [z_i]_{i=1}^{T}$, and W, b are parameters of a fully connected layer.

   As seen in Table 19, both balanced accuracy and F1 reduce for the models without sentiment (LSTM, GRU) indicating that integrating sentiment-wise sentences improves stance detection performance.

2. **Pooling Explanation** Popat et. al. find the most important phrases of input by removing phrases from the sequence and finding the ones with maximum effect on misclassification [92]. In our model, we can find the crucial information engaged in finding the stance of an argument using the max-pooling operation applied on the output of recurrent neural networks (see Section 4). We hypothesize that the more a token engaged in max-pooling, the more important the token is for final stance prediction. Below we find the most important words/phrases for a few questions in their arguments, that are correctly classified, with respect to their max-pooling engagement score, the frequency of the presence of a token in max-pooling operation. These phrase have the highest score across the argument. The words in [] have been added for clarification below and do not belong to the arguments.

    - Should students have to wear school uniforms? [NO] uniforms restrict students' freedom of expression
    - Are social networking sites good for our society? [NO] lead to stress and offline relationship
    - Should recreational marijuana be legal ? [NO] legalization, odious occasion
    - What are the pros and cons of milk's effect on cancer? [CON] dairy consumption is linked with rising death rates from prostate cancer
    - Is human activity responsible for climate change? [YES] significant because, (likely greater than 95 percent probability)
    - Is obesity a disease? [YES] no question that obesity is a disease, blood sugar is not functioning properly, disregulation, diabetes
    - Is the death penalty immoral? [YES] anymore, failed policy
    - Is drinking milk healthy for humans? [YES] contains many of those nutrients, excellent source of calcium, vitamin, riboflavin, pantothenic acid

    Figures 15, 16 show the heatmap plots of the two test instances. The numbers in each square is the frequency of the presence of a token in max-pooling operation. The darker colors

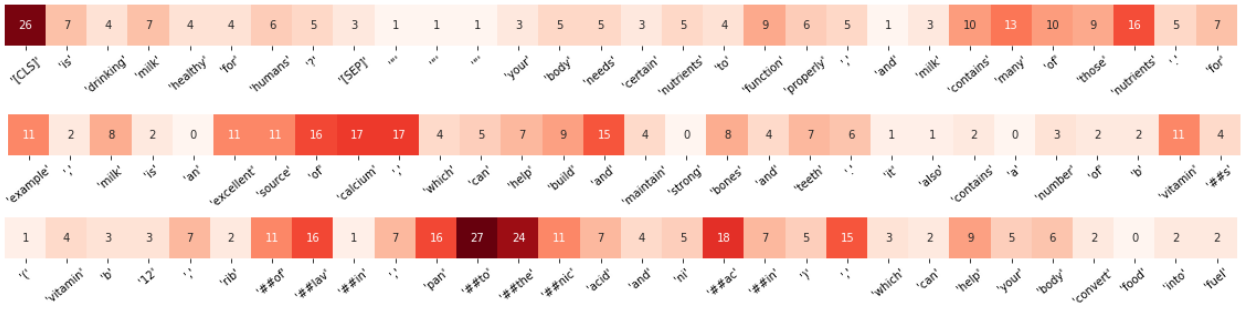


Figure 15: Heatmap of max-pooling matrix of a pro-opinion. Darker colors show larger scores

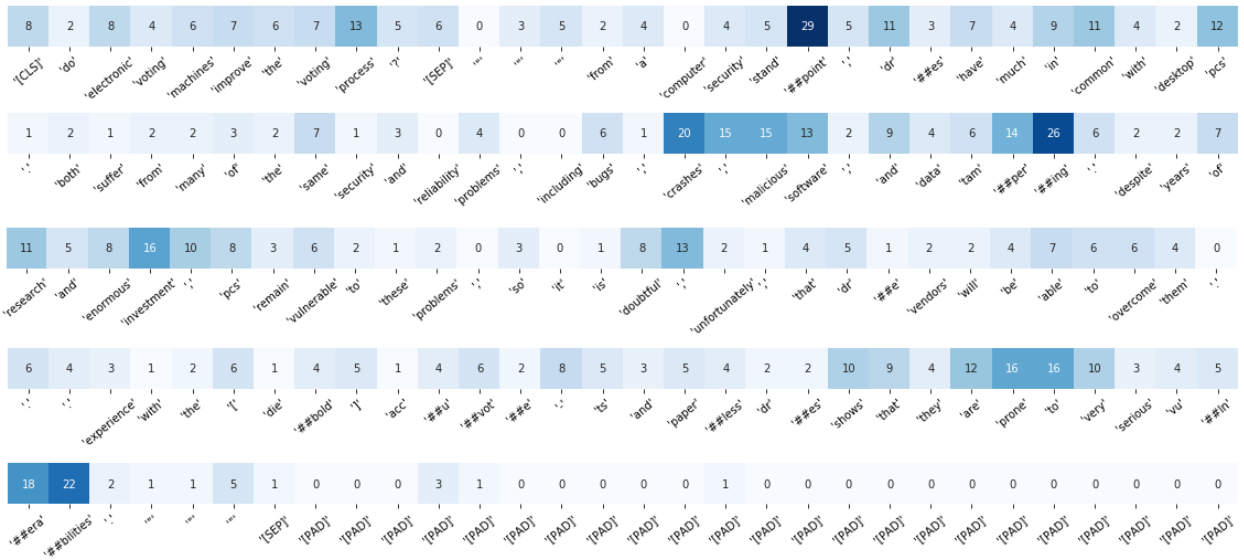


Figure 16: Heatmap of max-pooling matrix of a con-opinion. Darker colors show larger scores

explain how the model identifies the stance across the document against the question. The pro-example shows why the argument supports drinking milk well.

4.6 Conclusion

In this work, we study the problem of stance detection. In the first part, we propose a general model for stance detection of arguments. Unlike most models, our documents are long (with the average size of 166 words) and come from a large number of different domains. Experiments show promising results compared to the baselines. We also find our proposed model relies on the beginning of long arguments for stance detection. And depending on the discussed issue, sentiment of an argument

varies in pro or con class. Namely, pro-arguments express negative while con-argument have positive sentiment.

In the second part, we propose a model that leverages BERT representation with sentiment for stance detection. We create a new dataset for long fluid argument with 3X more instances compared to its previous version. The experiments on our benchmark dataset highlight the advantages of our approach. The model can reach BERT base parity with significantly fewer parameters. We also show how pooling operation explain the contribution of important phrases of arguments in detecting their stance against the given questions.

5 A GENERAL MODEL FOR AUTHORSHIP, STANCE AND HYPERPARTISAN DETECTION

With this study, we seek to investigate the effect of personality information in the three opinion-oriented tasks introduced earlier in this dissertation. We determine the personality of a given text's author as defined by the Myers-Briggs Type Indicators or MBTI [81]. Myers-Briggs uses four binary dimensions to classify people (Introvert–Extrovert, Intuitive–Sensing, Thinking–Feeling, Judging–Perceiving), e.g., INTJ, ENTJ, etc., getting 16 different types. The work allows us to dig into the personality space of authors from their writings on Web forums.

The personality signal even though noisy can be useful (as shown in this work) for downstream sentiment analysis, authorship verification, stance detection, and hyperpartisan news classification tasks as it is key information about the individual. Personality prediction does not only benefit commercial applications and psychology but also is advantageous in health care. Recent works find the relationship between personality types and social media behavior with depression and posttraumatic stress disorder [94]. Another research shows that certain personality types can anticipate mental illness and schizophrenia [73].

This problem poses a non-trivial challenge because a good solution must be capable of capturing the complexity and the depth of the human psyche as expressed through text. Anything short of that will result in task-specific pattern-matching. It follows that the main technical difficulty presented by the task at hand is the discrepancy between the corpora concerning their distributions, which results in the domain shift. This is where our work becomes relevant as it aims to bridge the gap by transfer learning and universal language understanding.

This problem is challenging because the human psyche is complex in its nature. The labels are fuzzy because the label distribution changes from population to population and the ground truth is not derived by the scientific method; rather, it is a set of ideas generally agreed upon by specialists and society. Furthermore, high quality curated datasets constructed by professional psychologists are difficult to obtain due to privacy reasons.

Models have only recently reached the capacity required. Our approach uses transfer learning through language understanding for personality prediction. We create a unified dataset from the collection of user inputs of three available MBTI datasets [30, 72, 91] originating from social media platforms including Reddit, Twitter, and Personality Cafe forum. We investigate how transfer learning with pretrained transformers contributes to personality prediction under a multi-label multi-class classification strategy. We analyze the relationship between personality types with the four specific tasks of stance, authorship, sentiment, and hyperpartisanship classification. The results on the unified dataset show that transfer learning along with pretrained bidirectional transformer models effectively changes Hamming loss, F1, and Jaccard Similarity for multi-label prediction (see Table 21). The contributions of current research are listed below:

- We propose a transformer-based model that utilizes the flow of sentiments across a document for Myer-Briggs personality type prediction. The model improves the BERT results on a large collection of three personality datasets.

- We show how transfer learning on our pre-trained personality model improves four downstream text classification tasks including sentiment analysis, authorship verification, stance detection, and hyperpartisanship detection. We, also, employ personality embeddings in non-transformer models such as convolutional neural networks for news data.

- Statistical analysis is provided to explain the effect of employment of the personality information in stance and hyperpartisan news classification.

Many naive personality prediction approaches rely on crafted features which can range from simple ones, such as TF-IDF of word or character n-grams to the ones produced by tools such as Linguistic Inquiry and Word Count (LIWC) [85], which extracts anything from low-level information such as Part Of Speech tags and topical preferences to psychological categories. These features are often aided by various psycholinguistic word lists that aim to detect emotions and sensory experiences [93].

5.1 Inducing Personality

Our work uses a bidirectional transformer to predict MBTI personality types using a large collection of data obtained from three existing personality datasets. Utilization of the pretrained word embeddings [71, 86] in many deep learning models indicates that leveraging knowledge obtained from unsupervised learning boosts the performance. Recently, language models pretrained on a large amount of raw text were shown to provide representations applicable to a wide variety of tasks with minimal fine-tuning [97, 40, 88]. These models can be effectively generalized to many downstream tasks and adapted to different domains. Below are the three main studies on utilizing online user-generated text for personality prediction. They are annotated with self-reported MBTI personality types of users.

- Reddit9K dataset is a large-scale dataset constructed from the posts and comments of 9K Reddit users. It is labeled with MBTI indicators and covers a wide variety of topics [30]. The authors extract user activity and linguistic features including word and character n-grams, LIWC word categories [85], and two Psycholinguistic dictionaries [93, 16]. Support Vector Machine (SVM), Logistic Regression (LR), and multi-layer perceptron are used to identify personality types and prove to be discriminative for personality prediction.

- Twitter Personality Prediction dataset is a large corpus of 1.2M tweets of 1.5K users [91]. Experiments performed by the dataset creators show that linguistic features are reliable representatives for two out of four personality dimensions. We hypothesize that the cause of the discrepancy is the difference between the distribution of personality types in social media users and the general U.S. population.

- Kaggle dataset collects the user posts of the Personality Cafe[15] forum and covers 8.6K different people with 16 MBTI personality types [72].

[15] https://www.personalitycafe.com/

5.2 MBTI vs. Big-5

Personality speaks of individual and stable differences in characteristic patterns of thinking, feeling, and behaving [17]. Myers-Briggs Theory (MBTI) and Big-5 are the two widely used personality theories, each designed for different reasons. Big-5 considers five personality traits including Openness, Conscientiousness, Extraversion, Agreeableness, and Neuroticism for humans. It finds how much a personality traits a person is and is useful for quantitative inference. It also measures a negative attribute, Neuroticism.

On the other hand, MBTI has four dichotomies including Extraversion/Introversion, how one gains energy, Sensing/iNtuition, how one processes information, Thinking/Feeling, how one makes decisions, Judging/Perceiving, how one presents herself to the outside world. It is a qualitative tool and categorizes people on the scale of 16 personality types. MBTI is designed for subjective tools and focuses on natural habits.

There are ongoing discussions about the expressivity and validity limitations of MBTI compared to Big-5 [90, 9]. Apparently, MBTI is receiving more criticism recently [110] as is becoming more popular among people for their career choice, companies for team building [1], and research studies as a tool [102, 122, 119]. Although the two theories are built on different perspectives, studies show that their dimensions to be correlated [26].

5.3 Data Limitation

Personality types can be acquired using a questionnaire or self-reported types. In NLP research some works use a questionnaire to annotate the text of blogs [44], emails [82], and essays [4], however, their data is not a suitable option for model generalization due to the small size, closed vocabulary, canonical language and low topic diversity. A large dataset that resolved the mentioned limitations on Facebook posts was created in 2012 [111], but is unavailable due to privacy issues. The aforementioned problems limit acquiring the ground truth data in personality and NLP research. So, utilizing the self-reported labels available on social media is our remaining option to

get a large personality dataset with open vocabulary, spontaneous language and various topics. Inevitably, these personality labels are noisy, but we expect that the statistical power of the large number of individuals from different social media platforms would reduce the effect of noisy labels in personality prediction. Next, we provide more details about the dataset.

5.4 Dataset

We use Reddit9k, Twitter Personality Prediction, and Kaggle Myers-Briggs Personality Type datasets to train and evaluate our proposed model for automatic personality type prediction. In all datasets, the annotation process relies on self-reported personality types, and no questionnaire is given to the users. Previously, *MyPersonality* created from Facebook user data was a questionnaire-based dataset. However, the owners of *MyPersonality* dataset [111] decided to stop sharing the dataset in 2018 and we are not able to use that. We make a unified dataset from the collection of the three available MBTI personality datasets and remove the non-English contents. Each instance in the original datasets is user posts in that social media platforms that could exceed up to 1M word tokens. So, we create shorter instances by splitting them into sequences of 512 word unigrams after cleaning and removing non-English contents. Our model is based on BERT and the maximum length of BERT input is 512 tokens. We find that the new dataset is highly skewed towards two out of four personality dimensions. There might be multiple reasons for that. I) According to [91] the distribution of personality types among the United States population is not balanced at all. II) Users from some specific personality types tend to participate in social media platforms and express their personality types more than others. Our experiments show that the class imbalance highly affects training, generating poor results for small classes, among evaluation methods. To alleviate the skewness of the data in training phase we can take two steps: adding class weights concerning their size in loss computation or making a balanced subset of the original dataset. We notice that the former does not improve the performance significantly, but, the latter shows to be a reliable solution. To create a balanced version of the dataset we over-sample small and under-sample large classes such that their final sizes are equal to the original average size of the 16 MBTI personality

Table 20: The number of instances in the unified personality dataset from Twitter, Reddit, and Personality Cafe forum; pers.:personality; # of dimensions: 4; # of types: 16

set	size	size/pers. type	size/pers. dimension
train	558,352	34,897	279,176
dev	79,776	4,986	39,888
test	159,520	9,970	79,760

types before sampling. Table 20 reports the unified dataset statistics after balancing.

5.5 Model

We build a general model to predict four MBTI personality dimensions. The MBTI dimensions are expressed as Booleans (0/1). The personality dimensions are I/E, or Introversion (I)/Extroversion (E); N/S, or Intuition (N)/Sensing (S); F/T, or Feeling (F)/Thinking (T); and J/P, or Judging (J)/Perceiving (P). Under this scheme, each instance can have multiple labels with four classes. The combination of these four classes gives $2^4 = 16$ MBTI personality types.

We consider multi-labeling instead of binary classification for MBTI personality prediction because of a huge difference between the size of the classes in binary scheme. Our experiments show that this difference causes the models to get skewed towards majority classes during training and it could not be handled by sampling techniques. Under sub-sampling, a tiny training set is created that leads to poor final results because the models could not differentiate the 16 classes with the tiny training set. On the other hand, over-sampling creates a huge dataset with hundreds of redundant examples. It leads to models failing in predicting the unseen examples of minority classes correctly because they get over-fitted by the redundant examples. However, we find that when we merge the 16 classes into four in the multi-labeling schema, overcoming the class imbalance becomes easier as the difference between the size of classes reduces significantly and we can handle class imbalanced by applying both over-sampling on small classes and sub-sampling on large classes simultaneously. Use of pre-trained language models and transformers shows significant improvements in various NLP problems. Recently, the bidirectional based transformer models such as BERT [18] and XLNet [120] overcome previously published language models trained on one direction (e.g. ULMFiT) [40] or

the shallow concatenation of left and right direction of text input (e.g. ELMo) [89] in various text classification tasks. We use BERT architecture as the basis of our personality prediction model. BERT takes position, segment, and token embedding as input to compute the importance of a token in a sequence. For personality classification purposes, we take into account the sentiment of sentences in an input sequence aside from the standard BERT input. According to [113], the level of emotion expression by people and the way they express their emotions define how people feel about the world. LIWC that organizes words in psychologically meaningful categories has been widely used in Psycholinguistics and is designed such that it can identify emotion in language use precisely [113].

We split the input sequence into linguistic sentences.[16] The sentiment of each sentence is one of positive, negative, or neutral, and is inferred using VADER sentiment score [43]. VADER is a rule-based model for the general sentiment analysis task. Each token inherits the sentiment of the sentence in which the token appears. For the two classifications ([CLS]) and separator [SEP] tokens, we use neutral embedding. Although VADER is a token-based sentiment tool, we use sentence-wise sentiment instead of token-wise because of two reasons: I) our intuition is to let the model learn the transition of sentiment across sentences, not tokens, and find one's personality when the sentiment of his/her opinion changes sentence to sentence. II) BERT uses sub-words units known as Word-pieces and each VADER lexicon may be composed of multiple Word-pieces. So, we must assign the sentiment of the VADER lexicon to all its Word-pieces eventually. For example, the sentiment of 'huggable' must be assigned to its three sub-words in our model: ['hug', 'ga', 'ble']. Also, our experiments on the dev set show that token-wise sentiment avoids learning the transition of sentiments and does not improve the model performance as much as sentence-wise sentiment.

The sentiments of input tokens will be embedded using a $3 \times k$ matrix that is randomly initialized where k is the size of the hidden states of the model. Then, these sentence-wise sentiment embeddings will be accumulated with the three standard embeddings of the BERT model ($E_t^{token}, E_t^{position}, E_t^{segment}$) to form the input embedding (E_t for token t). Figure 17 shows the

[16] we use NLTK sentence tokenizer [64].

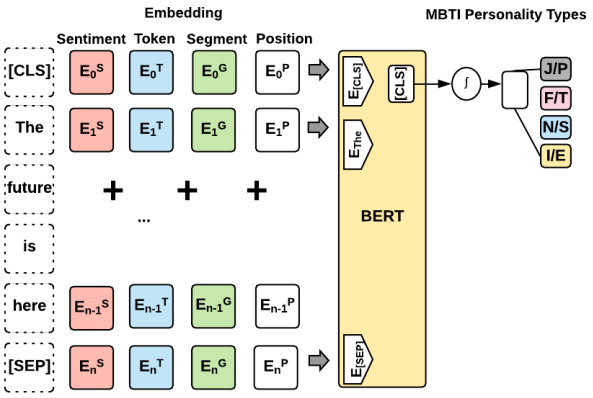


Figure 17: PersBERT model schema; Sum of four different embeddings forms the input.

model architecture.

$$E_t = E_t^{token} + E_t^{position} + E_t^{segment} + E_t^{sentiment} \tag{14}$$

The input embeddings will be given to the BERT sentence classification model that takes a sequence of linguistic sentences as one single input compared to the sentence-pair model that takes two inputs (e.g. a question and its answer). A fully connected layer forms a classifier that squeezes the pooled output (x) of the BERT model to four personality dimensions (I/E, N/S, F/T, and J/P). The hidden state of [CLS] token ($h_{[\text{cls}]}$) is used as the input of the pooling layer.

$$\begin{aligned} x &= \tanh(W^p h_{[\text{cls}]} + b^p), \\ \text{logit} &= W^c x + b^c \end{aligned} \tag{15}$$

where $W^c, W^p, b^p,$ and b^c are the layers' parameters.

Similar to other multi-label multi-class problems, the loss is the overall binary cross entropy among all classes.

$$L = \frac{1}{CN} \sum_{i \in N, c \in C} y_{c,i} \log \sigma(y'_{c,i}) + (1 - y_{c,i}) \log(1 - \sigma(y'_{c,i})) \tag{16}$$

where N is the number of examples, C number of classes, σ sigmoid function and y, y' are true labels and logits (input of probability function) respectively.

We refer to the proposed model as PersBERT for the rest of this research.

5.5.1 Multi-class Multi-label Baselines

We mentioned earlier that personality is connected to emotion and so sentiment. As its automatic prediction is being considered in multi-label mode, we choose the baselines that are widely used in sentiment analysis or multi-label classification. They are listed as following:

- Kim-CNN is one of the initial and successful applications of Convolutional Neural Network (CNN) for text classification [50].

- XML-CNN [63] is a CNN model for extreme multi-label text classification where the number of labels can exceed even a few thousand. The model architecture inherits Kim-CNN's model specification. Here, a dynamic max-pooling layer highlights important information across different parts of a document. XML-CNN was able to beat most of the deep learning baselines in six benchmark datasets.

- DocBERT is the BERT model with a fully connected layer that converts the hidden state of BERT pooling layer to C activations for C-class classification. The pooling layer pools the model by taking the hidden state corresponding to the classification token ([CLS]) of input sequence through non-linearity (tanh). We fine-tune DocBERT for classification and initialize it with pre-trained BERT-base-uncased weights. We do not fine-tune BERT under its pre-training schema.

- Hierarchical Attention Network (HAN) is a recurrent neural network model that mirrors the

hierarchical structure of the English language [121]. Applying attention mechanisms in word and sentence-level enables the model to find crucial parts of the document for the downstream classification task. The model outperforms its competitive baselines in sentiment analysis of user reviews dataset including Yelp, Amazon, and IMDB.

We examine CNN and LSTM models with two Google News (GNews) and FastText embeddings [70].

5.5.2 Evaluation

We develop our code based on the Hedwig[17] implementations of the baselines and train the models on 30 epochs with the batch size of 16 or 32. Training is controlled by early stopping with patience = 5 which will be stopped after 5 consequent epochs of no improvement of the highest F1 score gained. The test set is examined on the model with the best F1 of the dev set. We use all the settings for DocBERT and PersBERT same as the BERT-base-uncased model; the BERTAdam optimizer with the learning rate of $2e-5$ recommended in [18]; however, we set the sequence length = 256 for all models. Similar to DocBERT, all parameters of PersBERT are updated during backpropagation. Training PersBERT takes 5 days on a TITAN RTX GPU with batch size=16. For evaluating multi-label personality prediction, we use Hamming loss, Jaccard Similarity, F1-Macro, and F1-Micro scores. For more information about the measures please see [116]. We use scikit-learn [84] library for evaluation measures and other statistical methods.

Experimental results on the unified dataset (Table 20) are provided in Table 21. They indicate that PersBERT trained on 256 tokens of input sequence achieves the best results among baselines in all multi-class multi-label evaluation measures. An F1 improvement of $+0.5\%$ and -0.52% Hamming loss reduction on $\geq$ 159K test instances compared to DocBERT shows that adding sentiment embedding of sentences to the input distinguishes the personality types more accurately. Apart from that, both transformer-based models, DocBERT and PersBERT, show significant improvement over two CNN models with two different pre-trained embeddings and HAN (about +40% in

[17] https://github.com/castorini/hedwig

Table 21: Personality prediction: multi-class multi-label classification

Method	Set	Jaccard similarity	Hamming loss	Macro-F1	Micro-F1
Kim-CNN, GNews	dev	46.85	41.73	62.87	63.80
Kim-CNN, GNews	test	46.82	41.76	62.82	63.78
Kim-CNN, FastText	dev	45.94	39.20	62.33	62.96
Kim-CNN, FastText	test	45.83	39.31	62.23	62.86
XML-CNN, GNews	dev	44.68	45.83	56.60	61.77
XML-CNN, GNews	test	44.72	45.76	56.69	61.80
XML-CNN, FastText	dev	48.06	40.88	64.10	64.92
XML-CNN, FastText	test	47.97	40.96	64.0	64.83
HAN, GNews	dev	46.55	41.26	62.95	63.53
HAN, GNews	test	46.62	41.18	63.03	63.59
HAN, FastText	dev	46.27	38.55	62.82	63.27
HAN, FastText	test	46.29	38.48	62.83	63.29
DocBERT	dev	85.84	7.56	92.37	92.38
DocBERT	test	86.03	7.46	92.47	92.49
PersBERT	dev	**86.94**	**6.95**	**93.01**	**93.02**
PersBERT	test	**86.97**	**6.94**	**93.03**	**93.03**

Jacc. and -30% in Hamming loss). It should be noted that when we increase the class dimension and train the models in the binary classification strategy ($2^4 = 16$ MBTI classes) we usually get poor results compared to multi-label mode.

5.6 Transfer Learning

In classical Machine Learning approach, algorithms learn a task from one single dataset which needs a large set of data for an acceptable prediction. With transfer learning, algorithms leverage the data from various domains or tasks for a better generalization outcome [98]. As an example, pretrained world embeddings are one of the successful applications of the transfer learning methods in NLP [71]. In sequential transfer learning the tasks are learned in sequence of two stages. In the first stage, the pretraining, the model learns the general representations in the source data or task. This stage is followed by the adaptation in which the learned knowledge is applied and adapted on the target data. In this work we adopt sequential transfer learning to use the personality information in the three opinion-oriented tasks. Because we do not have personality types of stance, authorship,

and hyperpartisan data, we anticipate that the knowledge inferred from pretraining can be used for these tasks through sequential transfer learning. Accordingly, in pretraining phase the PersBERT model is trained for personality prediction. This model can be adapted for a new downstream task that is related to personality. There are two approaches for the adaptation phase. First, feature extraction, in which the weights of the model are frozen and directly used as the input of the target model. On the other hand, in fine-tuning approach all or part of the model parameters are fine-tuned and updated through the training phase of the target task or domain. Here, we, first take the fine-tuning approach for the adaptation as it has been found to accomplish better results than feature selection. Then, we report some experimental results using feature selection in Section 5.6.7.

So, we aim to study if the knowledge gained from the personality prediction model, PersBERT, helps other problems. Personality is closely connected with opinions and how people form opinions; hence, we choose the four tasks i.e., hyperpartisan news detection, sentiment analysis, authorship verification and stance detection that are designed around opinion mining (or reflect user behavior). We create our transfer learner model named DocBERT + PersBERT by sharing a fully connected layer between the two pretrained models; the personality model, PersBERT, and DocBERT (Figure 18). Our empirical experiments show that the transformer-based models achieve the best results among the baselines.

The shared layer, classifier layer, takes the concatenation of the output vector of pooling layers from the models (equation 17) and shrinks them to C classes. Recall that in DocBERT and PersBERT, the pooling layer pools the model by taking the hidden state corresponding to [CLS] token of input sequence using a non-linear activation (tanh).

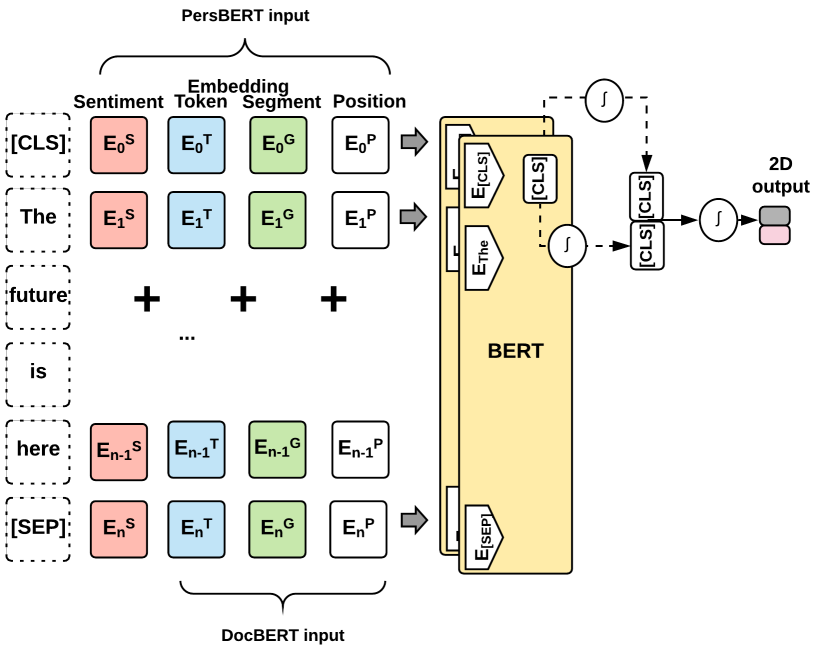


Figure 18: DocBERT+PersBERT architecture

$$
\begin{aligned}
x_{[\text{cls}]}^{\text{PersBERT}} &= \tanh(W^p h_{[\text{cls}]} + b^p), \\
x_{[\text{cls}]}^{\text{DocBERT}} &= \tanh(W^d h_{[\text{cls}]} + b^d), \\
x &= [x_{[\text{cls}]}^{\text{DocBERT}}; x_{[\text{cls}]}^{\text{PersBERT}}], \\
\text{logit} &= W^c x + b^c
\end{aligned}
\tag{17}
$$

where W^p, W^d, b^p, b^d are the parameters of PersBERT and DocBERT pooling layers. $[;]$ denotes concatenation, W^c is the $2 * k \times C$ classifier weight matrix and k is the size of the pooled vectors (hidden state). Finally, the classifier output, logit, is normalized with a Softmax function for downstream binary classification tasks.

Table 22: Hyperpartisan versus mainstream news

Hyperpartisan (right bias): Pro-life students at Wilfrid Laurier University say they will no longer be able to put up their pro-life flag display, with campus leaders explaining such exhibits are too traumatic after one last fall caused an uproar on the Canadian campus. The display used 10,000 pink and blue flags to represent the 100,000 abortions each year in Canada. Student Union President Tyler Van Herzele, in a letter shared campuswide, stated that "the visceral nature of the display interfered with the ability of some members of the Laurier community to safely attend class, travel to work or remain on campus to study."...
Mainstream (left-center bias): The Juvenile Probation Commission held an emergency meeting Monday that lasted long into the night and is reconvening again today to discuss Tucker's fate. RECOMMENDED VIDEO Tucker, who was promoted from assistant chief last summer after then-chief Jesse Williams quit, has been accused of ignoring gas leaks and fire hazards at Log Cabin Ranch for boys, not working hard enough to send youths to community rehabilitation programs rather than the juvenile hall lockup, and clashing with her staff...

5.6.1 Hyperpartisan News Detection

Partisanship is the quality or action of strongly supporting a person, principle or political party often without considering or judging the matter very carefully. Recently, the term "hyperpartisan news" is being used to define the extremely biased news in favor of the right or the left political party. The importance of detecting the hyperpartisanship in news has been increased recently, however, the high speed of news release needs an efficient automatic solution that detects the status of hyperpartisanship of articles. An example is given in Table 22.

SemEval 2019 task 4 proposes hyperpartisan news detection. It has released train and dev sets of two versions of the hyperpartisan news dataset. It should be noted that the test set is not publicly available yet. In the first version, news by-publisher, all articles are labeled by the overall bias of the publisher as provided by BuzzFeed[18] journalists or MediaBiasFactCheck.com while in news by-article dataset documents are labeled manually by the agreement of the journalists. More information about the labeling process can be found here [49].

The Hyperpartisan new by-publisher dataset is imbalanced for the number of articles per publisher. Some publishers have fewer articles because they delete or relocate the older articles in their

[18]https://www.buzzfeed.com/

Facebook feed or site maps. The dataset may contain some non-political articles and these articles might be removed after some modification by SemEval in the future. The dev dataset is created in the same way as the training set. However, no publisher of the train will be existed in the test and dev sets to avoid the classifiers fitting the publishers. Because the test set is not released yet, we use SemEval dev set as test and split its train set into new train and dev sets with no publisher in common. Likewise, for the by-article dataset SemEval training set is divided into new train and dev sets and the original dev set is used for test.

Our topic modeling analysis on the by-publisher train set reveals that it is highly imbalanced in terms of news classes. We use non-negative matrix factorization to estimate topic distribution in news. For some topics, top documents belong to only one class. To avoid the models to learn topics but hyperpartisanship we sample from the training set so that the resulting set includes even number of unique examples per topic. We only apply sub-sampling on the train set and keep the dev and test set intact. This process increased F1 of DocBERT model by 5% on dev set. Table 23 provides the dataset statistics.

5.6.2 Stance Detection

The problem of stance detection is to identify if an opinion supports an idea or contradicts it. We use the new version of Procon dataset [36] for stance detection. It covers the argumentative opinions of different controversial issues ranges from education and immigration to birth control. We split the dataset with $4,264$ instances into portions of $(70\%, 10\%, 20\%)$ for train, dev and test sets. As each instance in Procon dataset is a pair of a question about an issue and an opinion regarding that, we use BERT sentence-pair model for both DocBERT and DocBERT + PersBERT models. So the input of the two BERT-based models is formed as *[CLS] question [SEP] opinion [SEP]*.

Table 23: Hyperpartisan news dataset; Hyp.: Hyperpartisan, Main.: Mainstream

	by-article			by-publisher		
sets	Hyp.	Main.	total	Hyp.	Main.	total
train	142	244	386	82,472	82,472	164,944
val	48	81	129	59,974	58,536	118,510
test	48	82	130	74,927	74,867	149,794

5.6.3 Authorship Verification

Authorship Verification (AV) identifies whether a pair of documents are written by the same author or not. It is being used in plagiarism detection, forensic analysis and sockpuppet detection, to new a few. We examine our model on three standard PAN AV datasets [2] which were studied in Chapter 3. Each dataset contains one train and one test set. We split the original train set into two portions of $(70\%, 30\%)$ for training and dev sets and evaluate on the original test set. The datasets statistics can be found in Table 8 of Chapter 3. Similarly, each instance in the AV dataset is a pair of documents, so, we use BERT sentence-pair model for both DocBERT and DocBERT + PersBERT models. The input is formed as *[CLS] first document [SEP] second document [SEP]* where documents are written by one or two unknown author(s) and may contain several (linguistic) sentences.

5.6.4 Sentiment Analysis

For sentiment analysis, UCI-Yelp and Amazon datasets[19] containing 1000 reviews with binary (positive or negative) labels are used. We use the same version of the datasets as in [55] and split them into three train, dev, and test sets with the size of 70%,10%, and 20%.

5.6.5 Settings

Here, the settings, training strategy, and baselines are the same as PersBERT's. All parameters of the transformer models, including DocBERT, PersBERT and DocBERT + PersBERT are updated

[19]https://archive.ics.uci.edu/ml/datasets/Sentiment+Labelled+Sentences

Table 24: Effect of personality embedding in opinion-oriented tasks; P: Precision; R: Recall

		DocBERT			DocBERT+PersBERT		
Dataset	Task	P	R	F1	P	R	F1
UCI-Yelp	sentiment analysis	91.92	95.79	93.81	91.18	97.89	**94.42**
UCI-Amazon		91.51	93.27	92.38	93.33	94.23	**93.78**
Procon	stance detection	72.89	77.65	75.20	77.09	82.87	**79.87**
PAN 2014 Essay	authorship verification	65.09	69.70	67.32	62.41	83.84	**71.55**
PAN 2014 Novel		60.33	73.74	66.36	65.15	86.87	**74.46**
PAN 2015		44.26	75.60	55.83	59.11	68.80	**63.59**
News by-article	hyperpartisanship detection	77.27	72.34	74.73	84.62	70.21	**76.74**
News by-publisher		61.48	38.33	**47.22**	57.04	33.29	42.05

during backpropagation. For AV datasets with pairs of inputs, the overall length of the two input documents does not exceed 512 tokens while the maximum length of input for other datasets is 256.

5.6.6 Results and Analysis

Transfer learning results are provided in Table 24 for eight different datasets. We only report DocBERT results as baseline because it achieves much higher performance compared to other models for the benchmark datasets. In sentiment analysis task, DocBERT + PersBERT improves DocBERT's F1 score by 1.4% and by 0.61% for UCI-Amazon and Yelp test sets respectively. As the PersBERT model is pre-trained on the personality data, it shows that the model captures personality-oriented features from short reviews resulting in highlighting class differences for sentiment analysis. Similarly, for the stance detection task (Procon dataset) DocBERT + PersBERT beats DocBERT by more than 4.5% overall. We plot the distribution of 16 personality types among the pro and con classes for four different issues (Figures 19 and 20). The figures exhibits the large margin between the two stance classes with respect to their personality types. This margins shows why adding personality information to the BERT model results in more accurate differentiation of proponent and opponent arguments that results in final F1 improvement.

Results of all three authorship verification datasets also show a similar trend. DocBERT + PersBERT overcomes the BERT classification model by more than $4\%, 8\%$ and 7.7% in PAN2014

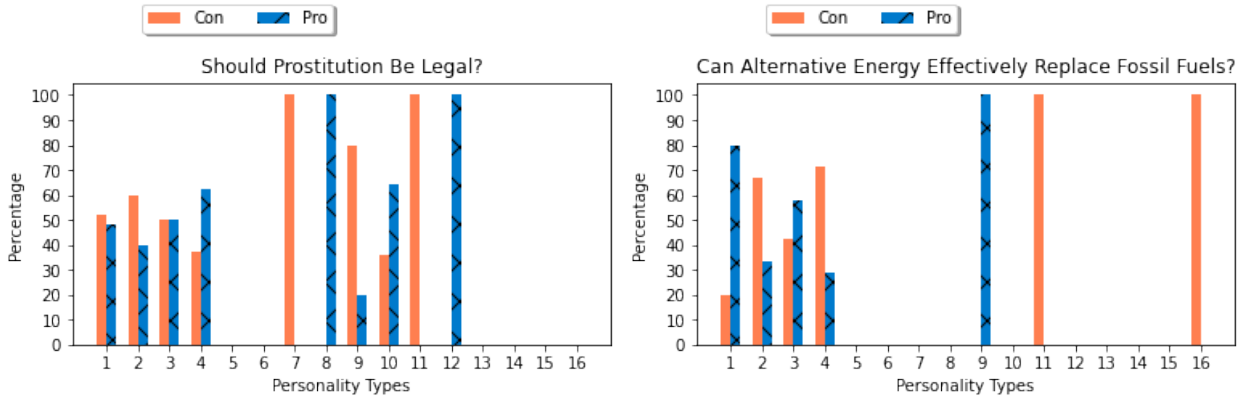


Figure 19: MBTI personality types distribution among proponents and opponents for different issues

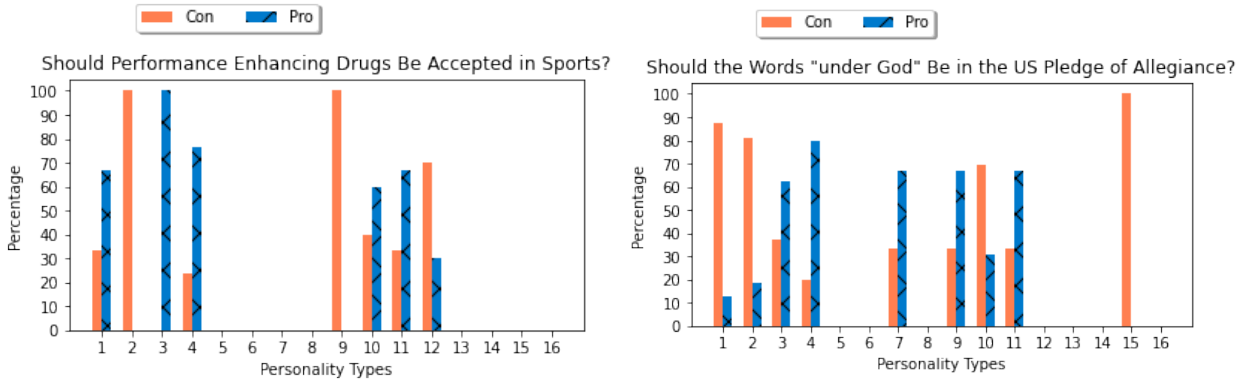


Figure 20: MBTI personality types distribution among proponents and opponents for different issues

Essay, PAN2014 Novel and PAN2015 respectively. Despite the improvements of F1 in news by-article results using personality information, we do not see the same effect on Hyperpartisan news by-publisher results. There is a reduction of -5.17% of F1 in news by-publisher when personality information is added. We plot MBTI personality distribution among hyperpartisan and mainstream news classes for news by-publisher datasets (Figure 21) to show the reason. Comparing the personality indicators distribution among the three train, dev, and test sets of the dataset in Figure 21 reveals that there is at most 5% difference in the size of personality indicators between hyperpartisan and mainstream news classes. Moreover, I/E, N/S, and J/P of the test set have almost equal size in the two news classes. Such similar personality distribution is not distinctive and unables the DocBERT+PersBERT model to differentiate the two news classes based on personality types precisely.

We hypothesize that there are a few main reasons for this: first, PersBERT uses sentiment features for personality prediction. However, we do not see any significant difference of VADER sentiment score between the two hyperpartisan and mainstream classes of the train set according to Figure 22. Secondly, the PersBERT model is trained on social media data while news data is formal and usually follows its publisher's strict regulations. It may lead to hiding the author's informal writing and personality features (Table 22). This difference between the language of news and social media data challenges the effect of transfer learning between the two domains. Finally, It is expected that personality distribution differs between mainstream and hyperpartisan classes for different news topics, similar to what we observed in stance detection.

The following section provides deeper experiments for hyperpartisanship detection.

5.6.7 A Deeper Look into Hyperpartisan News Detection and Personality

Table 24 shows that the employed approach was not useful in identifying hyperpartisanship in news by-publisher dataset. However, we anticipate there are some *connections* between personality and hyperpartisanship as opinion forms a bridge between these two concepts. So, we design the following experiments to investigate the hidden connection.

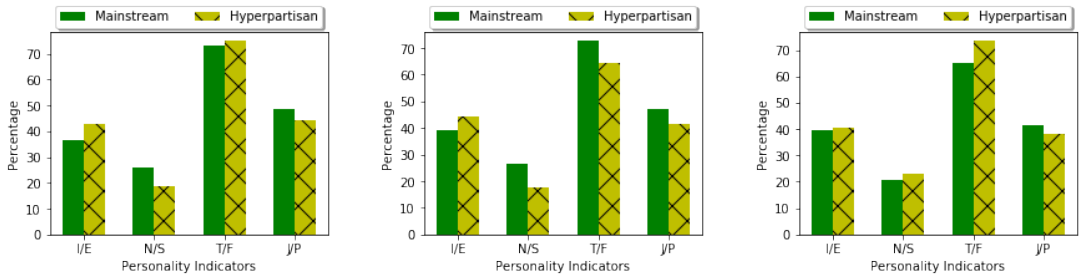


Figure 21: Train (left), dev (middle) and test set (right) news by-publisher distribution across MBTI personality dimensions. y-axis: percentage of each news class population, x-axis: personality dimensions

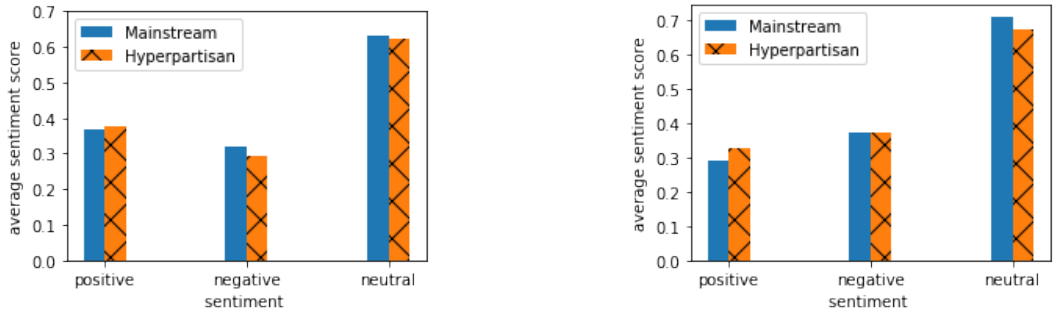


Figure 22: Sentiment distribution among hyperpartisan and mainstream news classes of by-publisher (left) and by-article (right) training sets

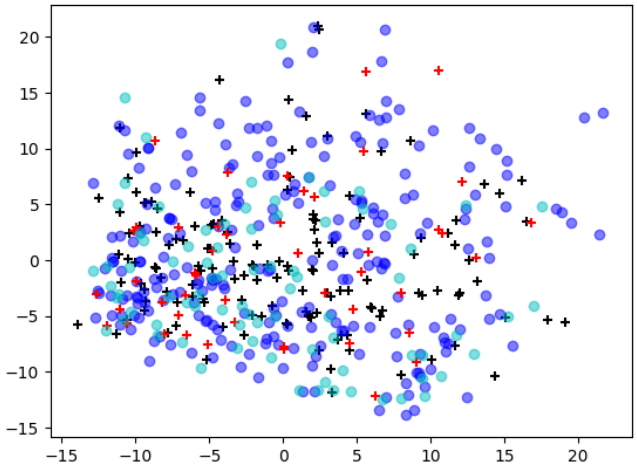


Figure 23: The projection of first two components of PCA for training and test sets of News by-article dataset. black +: hyperpartisan train, red +: hyperpartisan test, blue circle: mainstream train, cyan circle: mainstream test

Personality Features for News Classification Given news articles, we induce personality embedding using the proposed model, pre-trained on the personality dataset. We define the output of the PersBERT pooling layer of [CLS] token as personality embedding (PersBERT Emb) for feature-based classification (SVM). For neural networks models we use the respective hidden state of the last layer of the PersBERT model for each token. Here, the task is to investigate if personality embeddings provide any effective signal to differentiate the hyperpartisan and mainstream news articles in other models. The model take the induced personality embedding of news articles from PersBERT as input. We run different experiments on the two news datasets. Table 25 provides the results. According to the table, we observe that our model which is trained on the sequences of 256 tokens (half of the original BERT model) and social media data, contains more discriminatory features for classification than word-unigrams with 9673 features or even with top 80% χ^2 [20] features. It should be noted that if the whole document is used for feature extraction, the n-gram representation is among the best results for the small news by-article dataset [49]. Figure 23 shows the first two principal components of news by-article dataset. Moreover, XML-CNN gains better

[20] chi-Squared

Table 25: Effect of personality embedding on news datasets for sequence length=256; classifier: SVM-Linear kernel; b-unigram:binary-unigram; P:Precision; R:Recall

news by-article				news by-publisher			
Model	P	R	F1	Model	P	R	F1
SVM, b-unigram(9673)	58.62	36.17	44.74	SVM, 1-3gram	65.21	28.06	39.24
SVM, χ^2 b-uni[21](7738)	78.26	38.30	51.43	XMl-CNN, BERT Emb	63.09	36.80	46.48
SVM, PersBERT(768)	61.90	55.32	**58.43**	**XMl-CNN, PersBERT**	59.88	42.02	**49.38**

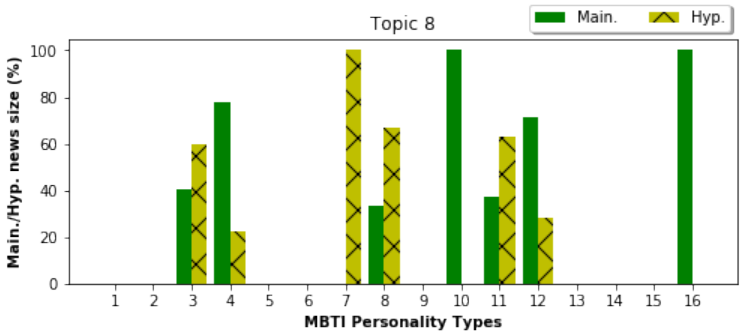


Figure 24: Personality distribution for mainstream/hyperpartisan news by-publisher training set of topic 8

performance on PersBERT embedding compared to the BERT on new by-publisher dataset.

Learning with Topic-Based Sub-Sampled Data News articles of the large by-publisher datasets cover a wide variety of topics. Inspiring by our analysis of personality distribution for different issues of stance detection task (Figures 19 and 20), we investigate if personality types differ in mainstream and hyperpartisan classes for different news topics. So, we first model the topics of news training set using the Non-negative Matrix Factorization (NMF) algorithm for 20 topics. Table 26 shows top 10 words for the two selected topics. Then, we choose distinct articles for each topic and use the personality model, introduced in section 5.5, to predict four MBTI personality dimensions. A tuple of four dimensions gives us an MBTI personality type. We plot 16 MBTI personality types versus two news classes for the two topics (Figures 24, 25). According to Figure 24, there is a remarkable difference between several personality types of the two news classes. 100% of personality type 7 and 10 is from hyperpartisan and mainstream news class respectively; or around 80% of personality type 4 belongs to mainstream while 70% of type 8 forms hyperpartisan

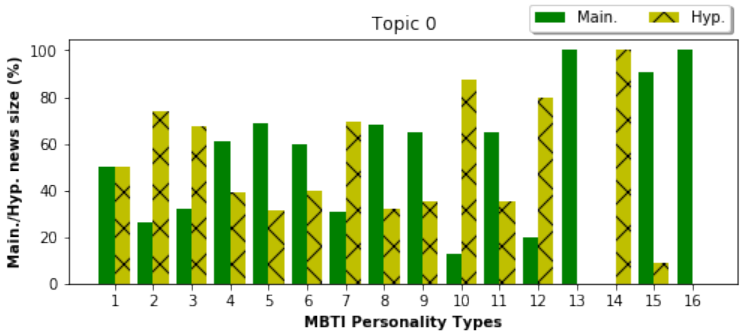


Figure 25: Personality distribution for mainstream/hyperpartisan news by-publisher training set of topic 0

Table 26: Effect of topic and personality based sub-sampling on hyperpartisan news detection; B.Acc.: balanced accuracy; dataset: news by-publisher dataset; training set: examples sub-sampled from topic modeling results; test and dev sets are the original sets; sl: sequence length (256 unless otherwise stated)

Training set	$\overline{\text{Entropy}}$	Model	Set	B.Acc.	top 10 words
topic 8	0.2795	DocBERT	dev	53.32	percent, rose, fell, index, rate, points, shares, economy, stocks, average
		DocBERT	test	50.02	
		DocBERT+PersBERT	dev	57.58	
		DocBERT+PersBERT	test	**55.51**	
		DocBERT, sl=512	test	50	
		DocBERT+PersBERT, sl=512	test	**58.28**	
topic 0	0.6837	DocBERT	dev	62.22	like, people, just don, know, time, women, life, think, way
		DocBERT	test	**52.82**	
		DocBERT+PersBERT	dev	55.59	
		DocBERT+PersBERT	test	50.15	

news. No news article of topic 8 belong to personality types 1, 2, 13, 14, and 15.

We report average entropy of the two news classes across all personality types to measure the difference of personality distributions between the two news classes (Table 26):

$$\overline{\text{Entropy}} = \frac{1}{|T|} \sum_{t \in T} \sum_{i \in I} -p_{i,t} \log_2(p_{i,t}) \tag{18}$$

where T is the set of all personality types, $I = [Hyp., Main.]$, and $|.|$ denotes the size, and $p_{i,t} = \frac{n_{i,t}}{\sum_{i \in I} n_{i,t}}$ is the proportion of news class i in personality class t. The smaller the entropy, the more the two news classes having different distribution (population) among 16 personality types.

Table 27: Hyperpartisan news by-publisher results with entropy-based sampling; *:p-value of McNemar's test $\leq 0.1 \times 10^{-4}$

		DocBERT			DocBERT+PersBERT		
Training data	$\overline{\text{Entropy}}$	**P**	**R**	**F1**	**P**	**R**	**F1**
all	0.9924	61.48	38.33	47.22	57.04	33.29	42.05
sub-sampled	$0.0999 \leq 0.1$	61.69	44.09	51.43	58.03	47.16	**52.04***

We train both DocBERT and DocBERT+PersBert on about 500 articles from topic 0 and topic 8, where there is a remarkable difference in the personality distribution among the news classes, separately and evaluate it on the original test set. Table 26 shows the results. According to the table, training on topic 8 with lower entropy results in higher balanced accuracy. This result shows that topic-based sub-sampling helps to shrink the news data to gives us a more distinctive representation of personality types that contributes in better hyperpartisan news detection. On the other hand, training on the data with higher entropy (topic 0) results in lower accuracy of DocBERT+PersBERT compared to DocBERT.

As a further experiment, we sub-sampled the whole training data such that the average entropy does not exceed the low amount of 0.1. At each sampling iteration, we choose an example from the original training data and add it to the sub-sampled set only if the $\overline{\text{entropy}}$ of the new set does not exceed the limit. The sampling gives us 111,614 training examples for $\overline{\text{entropy}} \leq 0.1$ (about 50K of the training examples are removed). Results in Table 27 reveal that training on data with unbalanced personality distribution of news classes results in remarkable improvements for both models with higher F1 of DocBERT+PersBERT compared to DocBERT.

5.7 Conclusion

We include sentiment embedding into BERT sentence classification architecture to predict MBTI personality dimensions. We use short sequences of online user posts gained from three major social network platforms for training. Our pretrained personality transformer improves BERT results in three PAN authorship verification, stance detection, and hyperpartisan news by-article datasets where the opinion is well-reflected.

Although personality information induced from our proposed model does not improve DocBERT results for hyperpartisan news by-publisher dataset at first, they can be used for an effective sub-sampling in hyparpartisanship detection. Moreover, personality embeddings can still compete with BERT embeddings in feature-based classification of small news by-article dataset and convolutional neural network models such as XML-CNN for the larger news by-publisher dataset.

6 Conclusion and Future Work

In this dissertation, we concentrate on content and stylistic techniques for three main opinion-oriented text classification tasks. The tasks investigate *who* the author of a text is, *which* stance one takes about a contemporary issue, and *is* the author's opinion biased towards left/right political views? We study the first task in online reviews, user-posts, machine learning articles, short essays, and novels. The choice of vocabulary (semantic) and grammar (syntax) are the two main distinctive features in one's writing style. However, traditional models that mostly rely on the choice of vocabulary are usually less successful in identifying the author of domain-independent document collections or deceptive contents. In the former case, the user's text originates from a wide variety of domains, a typical form of authorship identification in social media platforms, and in the latter case, the author hides his/her identity while spreading opinion spam.

We find that structural features are distinctive in individual's writing style and that they are the critical factors of leading more reliable results in both classical and deep neural network models. We deploy two approaches to utilize these syntactic features. In the first approach, a pre-trained statistical grammar parser is used to extract the structure of a sentence. The parser gives us a tree structure of a sentence that is traversed in four different ways to produce the structural features. We show that a feature selection method based on KL-Divergence finds discriminative features leading to better performance of the sockpuppet detection model. Moreover, relying solely on these structural features in our hierarchical attention, recurrent neural network outperforms several state-of-the-art models showing the effectiveness of these features in identifying writing style change detection.

We also deploy a universal pre-trained language model for stance detection, a case of opinion mining. In this task, the whole polarity of long argumentative opinions against a contemporary issue is determined. As the polarity of an argument can be expressed anywhere within it, and LSTM forget-gates may lose the long dependencies, max and average pooling operations avoid losing stance-related information over time. Our analysis of the tokens' engagement score in max-pooling

operation shows that the proposed language model-based architecture locates the informative words at the beginning of the long arguments, whereas they are discovered anywhere in shorter arguments. Universal language models, even pre-trained on uni-direction, show compelling results when fine-tuned on long, complicated arguments for stance detection.

In the last task, we model personality type as a new feature for opinion mining. Opinion associates with personality and can be used as an auxiliary feature. We believe that this study is the first work that investigates the connection between personality types and opinion-oriented tasks by inducing personality from text. With the recent improvements of bidirectional language models (transformers), we utilize them to infer personality traits from user's text to enhance the accuracy of opinion-oriented text classification models. Personality traits, even though noisy, can be inferred using language models if a large amount of data is available. This task is different from identifying direct signals from text (finding positive or negative words), a typical process in regular opinion mining applications such as stance detection or sentiment analysis. We collect a unified dataset from three available personality data gathered from various social media platforms. The labels are self-determined by users, and no questionnaire is used for labeling. Due to privacy issues, the only sizeable public personality dataset labeled with the help of the questionnaire became unavailable in 2018. We add sentiment embedding to the BERT model to predict Myers-Briggs personality types in the multi-label classification scheme. Results show that engaging sentiment improves personality prediction. We connect the BERT model with the pre-trained personality model and fine-tune it on the three main tasks in this study. We find personality information quite helpful in authorship verification and stance detection as connecting personality model to BERT outperforms the BERT without personality.

However, we do not see the same improvement for the hyperpartisan news detection that are labeled based on the publisher bias at first glance. We find that this result is aligned with the personality distribution of news training set because the personality model does not find a significant difference between the personality distribution of mainstream and hyperpartisan news articles. We hypothesize that there are a few reasons for this effect. First, news articles are formal and are written

concerning the strict publisher's regulations. It may result in diminishing the personality signals of the author when articulating. On the other hand, our personality model is trained on social media posts with informal and every day language that is immensely different from the news language. Another reason might be the similar personality types of authors from the mainstream and left/right news spectrum. However, experiments on entropy-based sub-sampled news training data revile new insights about personality distribution. Training the joint model (Personality+BERT) on a set with low entropy outperforms the test results of BERT model, trained on the same set, with a large margin. We should note that in all the above applications, the sole usage of the personality model does not supersede the original BERT model. Indeed, the personality information enriches the existing BERT model for a better downstream opinion-based classification task.

The current achievement of personality fingerprint induced from writings lead us to envision several perspectives of future works. On the application side, personality fingerprint can be studied in other NLP problems. As the personality model works effectively in authorship verification, it is expected to be helpful in other problems having a strong connection with authorship, including but not limited to academic fraud detection, deception detection, authorship attribution, and author profiling. Moreover, public opinions are being extensively studied in computational social science nowadays. For example, a research work analyzes a grassroots initiative to bring an Ohio anti-labor bill to a state-wide referendum by tracking group opinion of their natural conversation in social media platforms [10]. Other social science problems such as social phenomena or individuals' beliefs change upon receiving new information in social media are good candidates to be engaged with personality.

As a further step in modeling perspective, it is reasonable to explore the well-researched neural network architectures with fewer parameters compared to BERT designed for the specific NLP tasks to make the most of the personality information. For example, attention mechanisms, in conjunction with recurrent networks, have been actively used in opinion mining and authorship identification models and should be considered to be used with personality information.

Publications

- Marjan Hosseinia and Arjun Mukherjee, "Detecting Sockpuppets in Deceptive Opinion Spam", CICLing 2017, Budapest, Hungary.
- Marjan Hosseinia and Arjun Mukherjee, "Experiments with Neural Networks for Small and Large-Scale Authorship Verification", CICLing 2018, Hanoi, Vietnam.
- Marjan Hosseinia and Arjun Mukherjee, "A Parallel Hierarchical Attention Network for Style Change Detection", Working Notes of CLEF 2018, Avignon, France.
- Marjan Hosseinia, Eduard Dragut and Arjun Mukherjee, "Pro/Con: Neural Detection of Stance in Argumentative Opinions", SBP-BRiMS 2019 Proceedings, Washington DC, USA.
- Marjan Hosseinia, Eduard Dragut and Arjun Mukherjee "On Stance Detection using Bidirectional Transformers", under preparation.
- Marjan Hosseinia, Eduard Dragut, and Arjun Mukherjee "On the Usefulness of Personality Traits in Opinion-Oriented Tasks", under preparation.

Bibliography

[1] cpp, 2015.

[2] Pan datasets, 2015.

[3] Anand, P., Walker, M., Abbott, R., Fox Tree, J. E., Bowmani, R., and Minor, M. Cats rule and dogs drool!: Classifying stance in online debate. In *Proceedings of the 2nd Workshop on Computational Approaches to Subjectivity and Sentiment Analysis (WASSA 2.011)* (Portland, Oregon, June 2011), Association for Computational Linguistics, pp. 1–9.

[4] Argamon, S., Dhawle, S., Koppel, M., and Pennebaker, J. W. Lexical predictors of personality type. In *Proceedings of the 2005 Joint Annual Meeting of the Interface and the Classification Society of North America* (2005), pp. 1–16.

[5] Augenstein, I., Rocktäschel, T., Vlachos, A., and Bontcheva, K. Stance detection with bidirectional conditional encoding. *arXiv preprint arXiv:1606.05464* (2016).

[6] Bagnall, D. Author identification using multi-headed recurrent neural networks. *arXiv preprint arXiv:1506.04891* (2015).

[7] Banerjee, R., Feng, S., Kang, J. S., and Choi, Y. Keystroke patterns as prosody in digital writings: A case study with deceptive reviews and essays. *Empirical Methods on Natural Language Processing (EMNLP)* (2014).

[8] Bar-Haim, R., Bhattacharya, I., Dinuzzo, F., Saha, A., and Slonim, N. Stance classification of context-dependent claims. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers* (2017), vol. 1, pp. 251–261.

[9] Barbuto Jr, J. E. A critique of the Myers-Briggs type indicator and its operationalization of carl jung's psychological types. *Psychological Reports 80*, 2 (1997), 611–625.

[10] Brown, C. H. *Analyzing group behavior from language use with natural language processing and experimental methods: three applications in political science and sociology.* PhD thesis, University of Texas at Austin, Austin, Texas, 10 2018.

[11] Chang, C.-C., and Lin, C.-J. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology 2* (2011), 27:1–27:27. Software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm.

[12] Chapelle, O., and Zien, A. Semi-supervised classification by low density separation. In *AISTATS* (2005), pp. 57–64.

[13] Chen, W.-F., and Ku, L.-W. UTCNN: a deep learning model of stance classification on social media text. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers* (Osaka, Japan, Dec. 2016), The COLING 2016 Organizing Committee, pp. 1635–1645.

[14] Chen, W.-F., and Ku, L.-W. Utcnn: a deep learning model of stance classificationon on social media text. *arXiv preprint arXiv:1611.03599* (2016).

[15] Chomsky, N. *Syntactic structures*. Janua linguarum (Mouton, Paris).: Series Minor. Mouton and Co., The Hague, 1957.

[16] Coltheart, M. The mrc psycholinguistic database. *The Quarterly Journal of Experimental Psychology Section A 33*, 4 (1981), 497–505.

[17] Corr, P. J., and Matthews, G. *The Cambridge handbook of personality psychology*. Cambridge University Press New York, 2009.

[18] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[19] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. 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)* (Minneapolis, Minnesota, June 2019), Association for Computational Linguistics, pp. 4171–4186.

[20] Du, J., Xu, R., He, Y., and Gui, L. Stance classification with target-specific neural attention networks. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence* (2017), IJCAI'17, AAAI Press, p. 3988–3994.

[21] Ebrahimi, J., Dou, D., and Lowd, D. A joint sentiment-target-stance model for stance classification in tweets. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers* (2016), pp. 2656–2665.

[22] Fan, R.-E., Chang, K.-W., Hsieh, C.-J., Wang, X.-R., and Lin, C.-J. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research 9* (2008), 1871–1874. Software available at http://www.csie.ntu.edu.tw/ cjlin/liblinear/.

[23] Feng, S., Banerjee, R., and Choi, Y. Characterizing stylistic elements in syntactic structure. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learn-ing* (2012), Association for Compu-tational Linguistics, pp. 1522–1533.

[24] Feng, S., Banerjee, R., and Choi, Y. Syntactic stylometry for deception detection. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2* (2012), Association for Computational Linguistics, pp. 171–175.

[25] Frery, J., Largeron, C., and Juganaru-Mathieu, M. Ujm at clef in author verification based on optimized classification trees. In *CLEF 2014* (2014), p. 7p.

[26] Furnham, A. The big five versus the big four: the relationship between the Myers-Briggs type indicator (MBTI) and neo-pi five factor model of personality. *Personality and Individual Differences 21*, 2 (1996), 303–307.

[27] Fusilier, D. H., Montes-y Gómez, M., Rosso, P., and Cabrera, R. G. Detection of opinion spam with character n-grams. In *International Conference on Intelligent Text Processing and Computational Linguistics* (2015), Springer, pp. 285–294.

[28] Gamon, M. Linguistic correlates of style: authorship classification with deep linguistic analysis features. In *Proceedings of the 20th International Conference on Computational Linguistics* (2004), Association for Computational Linguistics, p. 611.

[29] Gilbert, C. H. E. Vader: A parsimonious rule-based model for sentiment analysis of social media text. In *Eighth International Conference on Weblogs and Social Media (ICWSM-14). Available at (20/04/16) http://comp. social. gatech. edu/papers/icwsm14. vader. hutto. pdf* (2014).

[30] Gjurković, M., and Šnajder, J. Reddit: A gold mine for personality prediction. In *Proceedings of the Second Workshop on Computational Modeling of People's Opinions, Personality, and Emotions in Social Media* (2018), pp. 87–97.

[31] Gokhman, S., Hancock, J., Prabhu, P., Ott, M., and Cardie, C. In search of a gold standard in studies of deception. In *Proceedings of the Workshop on Computational Approaches to Deception Detection* (2012), Association for Computational Linguistics, pp. 23–30.

[32] Graham, N., Hirst, G., and Marthi, B. Segmenting documents by stylistic character. *Natural Language Engineering 11*, 04 (2005), 397–415.

[33] Halvani, O., Winter, C., and Graner, L. On the usefulness of compression models for authorship verification. In *Proceedings of the 12th International Conference on Availability, Reliability and Security* (2017), ACM, p. 54.

[34] Hasan, K. S., and Ng, V. Stance classification of ideological debates: Data, models, features, and constraints. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing* (Nagoya, Japan, Oct. 2013), Asian Federation of Natural Language Processing, pp. 1348–1356.

[35] Hasan, K. S., and Ng, V. Why are you taking this stance? Identifying and classifying reasons in ideological debates. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (2014), pp. 751–762.

[36] Hosseinia, M., Dragut, E., and Mukherjee, A. Pro/con: Neural detection of stance in argumentative opinions. In *Social, Cultural, and Behavioral Modeling* (Cham, 2019), R. Thomson, H. Bisgin, C. Dancy, and A. Hyder, Eds., Springer International Publishing, pp. 21–30.

[37] Hosseinia, M., and Mukherjee, A. Detecting sockpuppets in deceptive opinion spam. In *International Conference on Computational Linguistics and Intelligent Text Processing* (2017), Springer, pp. 255–272.

[38] Hosseinia, M., and Mukherjee, A. Experiments with neural networks for small and large scale authorship verification. In *International Conference on Computational Linguistics and Intelligent Text Processing* (2018).

[39] Hosseinia, M., and Mukherjee, A. A parallel hierarchical attention network for style change detection : Notebook for PAN at CLEF 2018. In *CLEF (Working Notes)* (2018).

[40] Howard, J., and Ruder, S. 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)* (2018), vol. 1, pp. 328–339.

[41] Howard, J., and Ruder, S. 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, July 2018), Association for Computational Linguistics, pp. 328–339.

[42] Hutto, C. J., and Gilbert, E. Vader: A parsimonious rule-based model for sentiment analysis of social media text. In *ICWSM* (2014).

[43] Hutto, C. J., and Gilbert, E. Vader: A parsimonious rule-based model for sentiment analysis of social media text. In *Eighth International AAAI Conference on Weblogs and Social Media* (2014).

[44] Iacobelli, F., Gill, A. J., Nowson, S., and Oberlander, J. Large scale personality classification of bloggers. In *International Conference on Affective Computing and Intelligent Interaction* (2011), Springer, pp. 568–577.

[45] Japkowicz, N., Myers, C., Gluck, M., et al. A novelty detection approach to classification. In *IJCAI* (1995), vol. 1, pp. 518–523.

[46] Jindal, N., and Liu, B. Opinion spam and analysis. In *Proceedings of the 2008 International Conference on Web Search and Data Mining* (2008), ACM, pp. 219–230.

[47] Joachims, T. Transductive inference for text classification using support vector machines. In *ICML* (1999), vol. 99, pp. 200–209.

[48] Kestemont, M., Tschugnall, M., Stamatatos, E., Daelemans, W., Specht, G., Stein, B., and Potthast, M. Overview of the author identification task at PAN-2018: Cross-domain authorship attribution and style change detection. In *Working Notes Papers of the CLEF 2018 Evaluation Labs* (Sept. 2018), L. Cappellato, N. Ferro, J.-Y. Nie, and L. Soulier, Eds., CEUR Workshop Proceedings, CLEF and CEUR-WS.org.

[49] Kiesel, J., Mestre, M., Shukla, R., Vincent, E., Adineh, P., Corney, D., Stein, B., and Potthast, M. Semeval-2019 task 4: Hyperpartisan news detection. In *Proceedings of the 13th International Workshop on Semantic Evaluation* (2019), pp. 829–839.

[50] Kim, Y. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882* (2014).

[51] Klein, D., and Manning, C. D. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1* (2003), Association for Computational Linguistics, pp. 423–430. Software available at http://nlp.stanford.edu/software/lex-parser.shtml.

[52] Kochkina, E., Liakata, M., and Augenstein, I. Turing at semeval-2017 task 8: Sequential approach to rumour stance classification with branch-lstm. *arXiv preprint arXiv:1704.07221* (2017).

[53] Koppel, M., and Schler, J. Authorship verification as a one-class classification problem. In *Proceedings of the Twenty-first International Conference on Machine Learning* (2004), ACM, p. 62.

[54] Koppel, M., and Winter, Y. Determining if two documents are written by the same author. *Journal of the Association for Information Science and Technology 65*, 1 (2014), 178–187.

[55] Kotzias, D., Denil, M., De Freitas, N., and Smyth, P. From group to individual labels using deep features. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2015), ACM, pp. 597–606.

[56] Landauer, T. K., and Dumais, S. T. A solution to plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge. *Psychological Review 104*, 2 (1997), 211.

[57] Layton, R., Watters, P., and Dazeley, R. Authorship attribution for twitter in 140 characters or less. In *Cybercrime and Trustworthy Computing Workshop (CTC), 2010 Second* (2010), IEEE, pp. 1–8.

[58] Levy, R., Bilu, Y., Hershcovich, D., Aharoni, E., and Slonim, N. Context dependent claim detection. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers* (2014), pp. 1489–1500.

[59] Li, H., Chen, Z., Mukherjee, A., Liu, B., and Shao, J. Analyzing and detecting opinion spam on a large-scale dataset via temporal and spatial patterns. In *Ninth International AAAI Conference on Web and Social Media* (2015).

[60] Li, J., Ott, M., and Cardie, C. Identifying manipulated offerings on review portals. In *EMNLP* (2013), pp. 1933–1942.

[61] Li, J., Ott, M., Cardie, C., and Hovy, E. H. Towards a general rule for identifying deceptive opinion spam. In *ACL (1)* (2014), Citeseer, pp. 1566–1576.

[62] Lim, E.-P., Nguyen, V.-A., Jindal, N., Liu, B., and Lauw, H. W. Detecting product review spammers using rating behaviors. In *Proceedings of the 19th ACM international conference on Information and knowledge management* (2010), ACM, pp. 939–948.

[63] Liu, J., Chang, W.-C., Wu, Y., and Yang, Y. Deep learning for extreme multi-label text classification. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval* (2017), ACM, pp. 115–124.

[64] Loper, E., and Bird, S. Nltk: the natural language toolkit. *arXiv preprint cs/0205028* (2002).

[65] Lukasik, M., Srijith, P. K., Vu, D., Bontcheva, K., Zubiaga, A., and Cohn, T. Hawkes processes for continuous time sequence classification: an application to rumour stance classification in twitter. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)* (Berlin, Germany, Aug. 2016), Association for Computational Linguistics, pp. 393–398.

[66] Luyckx, K., and Daelemans, W. Authorship attribution and verification with many authors and limited data. In *Proceedings of the 22nd International Conference on Computational Linguistics-Volume 1* (2008), Association for Computational Linguistics, pp. 513–520.

[67] Manevitz, L., and Yousef, M. One-class document classification via neural networks. *Neurocomputing 70*, 7 (2007), 1466–1481.

[68] Merity, S., Keskar, N. S., and Socher, R. Regularizing and optimizing lstm language models. *arXiv preprint arXiv:1708.02182* (2017).

[69] Mikolov, T. Statistical language models based on neural networks. *Presentation at Google, Mountain View, 2nd April* (2012).

[70] Mikolov, T., Grave, E., Bojanowski, P., Puhrsch, C., and Joulin, A. Advances in pre-training distributed word representations. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)* (2018).

[71] Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems* (2013), pp. 3111–3119.

[72] Mitchell. Myers-Briggs personality type dataset. https://www.kaggle.com/datasnaek/mbti-type, 2017. [Online; accessed 2019-11-19].

[73] Mitchell, M., Hollingshead, K., and Coppersmith, G. Quantifying the language of schizophrenia in social media. In *Proceedings of the 2nd Workshop on Computational Linguistics and Clinical Psychology: From Linguistic Signal to Clinical Reality* (2015), pp. 11–20.

[74] Modaresi, P., and Gross, P. A language independent author verifier using fuzzy c-means clustering. In *CLEF (Working Notes)* (2014), pp. 1084–1091.

[75] Mohammad, S., Kiritchenko, S., Sobhani, P., Zhu, X., and Cherry, C. Semeval-2016 task 6: Detecting stance in tweets. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)* (2016), pp. 31–41.

[76] Mohammad, S. M., Sobhani, P., and Kiritchenko, S. Stance and sentiment in tweets. *ACM Transactions on Internet Technology (TOIT) 17*, 3 (2017), 26.

[77] Mohtarami, M., Baly, R., Glass, J., Nakov, P., Màrquez, L., and Moschitti, A. Automatic stance detection using end-to-end memory networks. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)* (New Orleans, Louisiana, June 2018), Association for Computational Linguistics, pp. 767–776.

[78] Mukherjee, A., Kumar, A., Liu, B., Wang, J., Hsu, M., Castellanos, M., and Ghosh, R. Spotting opinion spammers using behavioral footprints. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2013), ACM, pp. 632–640.

[79] Mukherjee, A., Liu, B., and Glance, N. Spotting fake reviewer groups in consumer reviews. In *Proceedings of the 21st International Conference on World Wide Web* (2012), ACM, pp. 191–200.

[80] Mukherjee, A., Venkataraman, V., Liu, B., and Glance, N. S. What yelp fake review filter might be doing? In *ICWSM* (2013).

[81] Myers, I. B., and Myers, P. B. *Gifts differing: Understanding personality type*. Nicholas Brealey, 2010.

[82] Oberlander, J., and Gill, A. J. Language with character: A stratified corpus comparison of individual differences in e-mail communication. *Discourse Processes 42*, 3 (2006), 239–270.

[83] Ott, M., Choi, Y., Cardie, C., and Hancock, J. T. Finding deceptive opinion spam by any stretch of the imagination. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1* (2011), Association for Computational Linguistics, pp. 309–319.

[84] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research 12* (2011), 2825–2830.

[85] Pennebaker, J. W., Boyd, R. L., Jordan, K., and Blackburn, K. *The development and psychometric properties of LIWC2015*. University of Texas at Austin, Austin, Texas, 2015.

[86] Pennington, J., Socher, R., and Manning, C. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (2014), pp. 1532–1543.

[87] Peters, M., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)* (New Orleans, Louisiana, June 2018), Association for Computational Linguistics, pp. 2227–2237.

[88] Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. Deep contextualized word representations. In *Proc. of NAACL* (2018).

[89] Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365* (2018).

[90] Pittenger, D. J. Measuring the mbti... and coming up short. *Journal of Career Planning and Employment 54*, 1 (1993), 48–52.

[91] Plank, B., and Hovy, D. Personality traits on twitter—or—how to get 1,500 personality tests in a week. In *Proceedings of the 6th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis* (2015), pp. 92–98.

[92] Popat, K., Mukherjee, S., Yates, A., and Weikum, G. Stancy: Stance classification based on consistency cues. *ArXiv abs/1910.06048* (2019).

[93] Preoţiuc-Pietro, D., Carpenter, J., and Ungar, L. Personality driven differences in paraphrase preference. In *Proceedings of the Second Workshop on NLP and Computational Social Science* (2017), pp. 17–26.

[94] Preoţiuc-Pietro, D., Eichstaedt, J., Park, G., Sap, M., Smith, L., Tobolsky, V., Schwartz, H. A., and Ungar, L. The role of personality, age, and gender in tweeting about mental illness. In *Proceedings of the 2nd Workshop on Computational Linguistics and Clinical Psychology: From Linguistic Signal to Clinical Reality* (2015), pp. 21–30.

[95] Qian, T., and Liu, B. Identifying multiple userids of the same author. In *EMNLP* (2013), pp. 1124–1135.

[96] Qian, T., Liu, B., Chen, L., Peng, Z., Zhong, M., He, G., Li, X., and Xu, G. Tri-training for authorship attribution with limited training data: a comprehensive study. *Neurocomputing 171* (2016), 798–806.

[97] Radford, A., Narasimhan, K., Salimans, T., and Sutskever, I. Improving language understanding by generative pre-training. *URL https://s3-us-west-2. amazonaws. com/openai-assets/research-covers/languageunsupervised/language understanding paper. pdf* (2018).

[98] Ruder, S., Peters, M. E., Swayamdipta, S., and Wolf, T. Transfer learning in natural language processing. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Tutorials* (Minneapolis, Minnesota, June 2019), Association for Computational Linguistics, pp. 15–18.

[99] Sanderson, C., and Guenter, S. Short text authorship attribution via sequence kernels, markov chains and author unmasking: An investigation. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing* (2006), Association for Computational Linguistics, pp. 482–491.

[100] Sapkota, U., Bethard, S., Montes-y Gómez, M., and Solorio, T. Not all character n-grams are created equal: A study in authorship attribution. In *Human Language Technologies: The 2015 Annual Conference of the North American Chapter of the ACL* (2015), pp. 93–102.

[101] Seroussi, Y., Bohnert, F., and Zukerman, I. Authorship attribution with author-aware topic models. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2* (2012), Association for Computational Linguistics, pp. 264–269.

[102] Smith, L. J., Curtis, C. P., Perry, M., LoVasco, L., Yorke, A. M., and Talley, S. A. Mbti® type and interprofessional communication in doctor of physical therapy students. *Internet Journal of Allied Health Sciences and Practice 17*, 4 (2019), 9.

[103] Sobhani, P., Inkpen, D., and Zhu, X. A dataset for multi-target stance detection. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers* (Valencia, Spain, Apr. 2017), Association for Computational Linguistics, pp. 551–557.

[104] Solorio, T., Hasan, R., and Mizan, M. A case study of sockpuppet detection in wikipedia. In *Workshop on Language Analysis in Social Media (LASM) at NAACL HLT* (2013), pp. 59–68.

[105] Solorio, T., Hasan, R., and Mizan, M. Sockpuppet detection in wikipedia: A corpus of real-world deceptive writing for linking identities. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)* (Reykjavik, Iceland, may 2014), European Language Resources Association (ELRA).

[106] Somasundaran, S., and Wiebe, J. Recognizing stances in ideological on-line debates. In *Proceedings of the NAACL HLT 2010 Workshop on Computational Approaches to Analysis and Generation of Emotion in Text* (Los Angeles, CA, June 2010), Association for Computational Linguistics, pp. 116–124.

[107] Sridhar, D., Foulds, J., Huang, B., Getoor, L., and Walker, M. Joint models of disagreement and stance in online debate. 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, July 2015), Association for Computational Linguistics, pp. 116–125.

[108] Stamatatos, E. A survey of modern authorship attribution methods. *Journal of the American Society for information Science and Technology 60*, 3 (2009), 538–556.

[109] Stamatatos, E., Rangel, F., Tschuggnall, M., Kestemont, M., Rosso, P., Stein, B., and Potthast, M. Overview of PAN-2018: Author Identification, Author Profiling, and Author Obfuscation. In *Experimental IR Meets Multilinguality, Multimodality, and Interaction. 9th International Conference of the CLEF Initiative (CLEF 18)* (Berlin Heidelberg New York, Sept. 2018), P. Bellot, C. Trabelsi, J. Mothe, F. Murtagh, J. Nie, L. Soulier, E. Sanjuan, L. Cappellato, and N. Ferro, Eds., Springer.

[110] Stein, R., and Swan, A. B. Evaluating the validity of Myers-Briggs type indicator theory: A teaching tool and window into intuitive psychology. *Social and Personality Psychology Compass 13*, 2 (2019), e12434.

[111] Stillwell, D. J., and Kosinski, M. mypersonality project: Example of successful utilization of online social networks for large-scale social research. *International Conference on Mobile Systems (MobiSys)* (2012).

[112] Sun, Q., Wang, Z., Zhu, Q., and Zhou, G. Stance detection with hierarchical attention network. In *Proceedings of the 27th International Conference on Computational Linguistics* (Santa Fe, New Mexico, USA, Aug. 2018), Association for Computational Linguistics, pp. 2399–2409.

[113] Tausczik, Y. R., and Pennebaker, J. W. The psychological meaning of words: Liwc and computerized text analysis methods. *Journal of Language and Social Psychology 29*, 1 (2010), 24–54.

[114] Vapnik, V. *The nature of statistical learning theory*. Springer Science & Business Media, 2013.

[115] Weinberger, K. Q., Blitzer, J., and Saul, L. K. Distance metric learning for large margin nearest neighbor classification. In *Advances in Neural Information Processing Systems* (2005), pp. 1473–1480.

[116] Wu, X.-Z., and Zhou, Z.-H. A unified view of multi-label performance measures. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70* (2017), JMLR. org, pp. 3780–3788.

[117] Xie, S., Wang, G., Lin, S., and Yu, P. S. Review spam detection via temporal pattern discovery. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2012), ACM, pp. 823–831.

[118] Xu, X., Li, W., Xu, D., and Tsang, I. Co-labeling for multi-view weakly labeled learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence PP*, 99 (2015), 1–1.

[119] Yang, C., Richard, G., and Durkin, M. The association between Myers-Briggs type indicator and psychiatry as the specialty choice. *International Journal of Medical Education 7* (2016), 48.

[120] Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R., and Le, Q. V. Xlnet: Generalized autoregressive pretraining for language understanding. *arXiv preprint arXiv:1906.08237* (2019).

[121] Yang, Z., Yang, D., Dyer, C., He, X., Smola, A., and Hovy, E. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (2016), pp. 1480–1489.

[122] Zardouz, S., German, M. A., Wu, E. C., and Djalilian, H. R. Personality types of otolaryngology resident applicants as described by the Myers-Briggs type indicator. *Otolaryngology–Head and Neck Surgery 144*, 5 (2011), 714–718.