A novel approach to stance detection in social media tweets by fusing ranked lists and sentiments

Abdulrahman I. Al-Ghadir[a], Aqil M. Azmi[a,*], Amir Hussain[b]

[a] *Department of Computer Science, College of Computer & Information Sciences, King Saud University, Riyadh 11543, Saudi Arabia*
[b] *School of Computing, Edinburgh Napier University, Edinburgh EH10 5DT, UK*

Abstract

Stance detection is a relatively new concept in data mining that aims to assign a stance label (favor, against, or none) to a social media post towards a specific pre-determined target. These targets may not be referred to in the post, and may not be the target of opinion in the post. In this paper, we propose a novel enhanced method for identifying the writer's stance of a given tweet. This comprises a three-phase process for stance detection: (a) tweets preprocessing; here we clean and normalize tweets (e.g., remove stop-words) to generate words and stems lists, (b) features generation; in this step, we create and fuse two dictionaries for generating features vector, and lastly (c) classification; all the instances of the features are classified based on the list of targets. Our innovative feature selection proposes fusion of two ranked lists (top-k) of term frequency–inverse document frequency (*tf-idf*) scores and the sentiment information. We evaluate our method using six different classifiers: K nearest neighbor (K-NN), discernibility-based K-NN, weighted K-NN, class-based K-NN, exemplar-based K-NN, and Support Vector Machines. Furthermore, we investigate the use of Principal Component Analysis and study its effect on performance. The model is evaluated on the benchmark dataset (SemEval-2016 task 6), and the results significance is determined using t-test. We achieve our best performance of macro F-score (averaged across all topics) of 76.45% using the weighted K-NN classifier. This tops the current state-of-the-art score of 74.44% on the same dataset.

Keywords: Stance detection, Sentiment analysis, Top-k, K-NN variants, Support vector machines, Twitter

1. Introduction

Stance detection is an automatic process of determining whenever the author is likely in favor, against, or neutral towards a proposition or target within the text [1, 2]. The target could either be a person, an organization, a policy, or a movement, etc. The field of stance detection is, undoubtedly, a critical one given the role that social media is playing in modern society and the power it has over influencing various aspects in society [3]. It is meant to gather the information that could be potentially useful in decision-making by individuals, organizations, or even at the governmental level. With the skyrocketing popularity of social media sites, mining information from such sites is only natural, given the amount of data that the users post. In a stance detection system, we in a way, determine the author's favorability towards a given target, which may not be explicitly mentioned in the text. Upon this capability, we can enhance an individual's profile based on his or her stance on different issues.

The task of detecting the stance has a long tradition in the domain of political and ideological online debates [4]. It also has been an important preprocessing step in combating fake news [5, 6]. There are other applications that may benefit from the automatic stance detection, including information retrieval, text summarization, opinion summarization [7], rumor verification [8], etc.

We can formulate the task of stance detection as follows: given a tweet text and a target entity (e.g., person or organization), determine whether the author of the tweet is in favor, against, or indifferent to the given target. The last case includes the possibility that we failed to infer. Table 1 lists sample examples of target-tweet pairs and the stance.

It ought to be noted that the field of stance detection is a challenging one [9]. Stance detection can be used in identifying the stance within posts and comments that are made on popular social media and microblogs sites, such as Facebook and Twitter. Since the posts that people write on these forums are often brief and are directed toward a specific audience who will understand the context, it is not always a simple matter for an outsider to determine if the author feels in favor, against it, or is neutral toward the topic about which they are writing [10].

The recent surge in popularity of stance detection follows the competition SemEval-2016 task 6 [4]. This prevalence follows the increase in the popularity of social media

*Corresponding author: Tel.: +966 (0)11 467-6574, Fax: +966 (0)11 467-5423.
Email addresses: abdulrahman.alghadir@gmail.com (Abdulrahman I. Al-Ghadir), aqil@ksu.edu.sa (Aqil M. Azmi), a.hussain@napier.ac.uk (Amir Hussain)

Table 1: Examples of target-tweet pairs and the stance. A stance has three possible values, pro (in favor), con (against), or none.

Target	Tweet	Stance
Legalization of abortion	Why is bacteria considered life on Mars, but a heartbeat is not considered life on earth? #heartbeat #SemST	Against
Climate change is a concern	We need to work with confidence, transparency and guided by consensus @manupulgarvidal at @UN_PGA event on #action2015 #SemST	None
Hillary Clinton	If you're not watching @HillaryClinton's speech right now you're missing her drop tons of wisdom. #SemST	Favor

platforms in the last decade or so. It is the same as sentiment analysis, a research area that has been around for a while, and whose growth follows that of the social media. Though sentiment analysis is an important aspect in the stance detection, we should not confuse between them. Even though both are related, they, however, look differently at the same thing, e.g. the tweet.

Stance and sentiment analysis are sub-fields of natural language processing (NLP). Both sub-fields share similar connotations, and they both involve some common NLP methodologies (e.g., feature extraction and classification) to achieve the goal of determining the stance or sentiment of any message or tweet accurately. In sentiment analysis, we are interested in knowing whether a piece of text is positive, negative, or neutral based only on the language used. Typically, it reflects the sentiment of the text, whether it's an overall polarity or target-dependent [11]. The stance, on the other hand, is defined for a target topic and can be independent of the sentiment of the language used [7]. A prime difference between stance classification and traditional aspect-level sentiment classification (except for target-dependent sentiment classification) is that the identification of stance is dependent on target which might not be explicitly uttered in the actual text [12].

In this work, we are targeting tweets, and the fact is, tweets are short text. We do not have enough contextual information within the text, and this poses a great challenge in detecting the stance given that it does not require the explicit presence of the target. Just to illustrate, suppose we have the following two tweets [13], "We don't inherit the earth from our parents we borrow it from our children", and "Last time I checked, Al Gore is a politician, not a scientist". For most humans, it is obvious that both tweets are related to the topic of climate change, and that each tweet expresses a particular stance towards the topic. But, to a machine, detecting this stance is problematic. The unstructured syntax of tweets along with the fact that machines lack proper contextual awareness and historical knowledge which humans have (e.g., who Al Gore is), makes this a challenging problem for the learning algorithms.

During the last few years, deep-learning (DL) has been all the hype. No doubt it does a beautiful job, but at the expense of the size of the dataset. When the dataset is small, DL performance degrades considerably. There is a huge effort behind preparing dataset, specially the annotation, which is mostly a manual task. That is why most of the datasets are small in nature, including the benchmark dataset used for the stance detection. We noted that many state-of-the-art works that relied on handcraft-based features had better luck with the performance than those using DL for detecting the stance. For instance, the proposed system with a handcraft-based feature in [14] outperformed the deep-learning approach in [15] using the same dataset.

Top-k is a popular technique for ranked retrieval. Given a constant k, and a ranking function f, a top-k query returns the k highest ranked objects according to f. It is most popular in database retrieval, but less so for solving NLP tasks. For instance, Elfardy and Diab [16] used top-k with *tf-idf* as a scoring function in their traditional linguistic features group for detecting the stance. For their task of identifying the gender of the posts' author, Al-Ghadir and Azmi [17] used a combination of top-k, stop-word removal, stemming, *tf-idf* etc. The scheme did well based on its performance compared to the state-of-the-art.

We may summarize our contribution in this work as follows:

- We propose an enhanced fusion of two top-k based ranked lists and sentiment information for identifying the writer's stance from a given tweet.
- For classification, we use support vector machines (SVM) along with five different variants of K nearest neighbor (K-NN), and optionally with a dimensionality reduction method (principal component analysis, or PCA).
- We show our system outperforms other existing systems by evaluating it on a standard benchmark dataset. We use a statistical test to confirm the significance of our evaluation.
- We show that the inclusion of sentiment information had an insignificant (statistically) impact on our stance detection system.

For the rest of the paper we will be using top-k and ranked list interchangeably. With our intention of using top-k to detect the stance, we decided to adapt the techniques used in [17] for the task at hand. We did not go

for [16] since its performance was not so stellar (Table 4, entry CU-GWU). Moreover, [17] was used for a slightly different task with good performance, so it is worth seeing how it will fair with our problem of stance detection. Part of the adaption involved: (a) removing the feature "frequent characters", and (b) introducing spelling correction and stop-word removal as part of the preprocessing phase. The feature "frequent characters" was defined in a way that suited the relatively large size of the social forums used in the study. However, it is ill suited for tweets, which are much shorter than the posts, the object of our stance detection. In fact, using this feature for stance detection degraded the classification accuracy. In our investigation we noted that adding spell-correction and stop-word removal to the preprocessing phase helped increase the precision of the frequencies in the dictionaries, which in turn boosted the classification accuracy.

More specifically, our feature selection fuses two ranked lists along with the sentiment information. The ranked lists are *tf-idf* scores from fusing two dictionaries for words and stems. For classification, we use the standard K-NN and four of its variants. These are: discernibility-based K-NN, weighted K-NN, class-based K-NN, and exemplar-based K-NN. As we will see later (Section 2), none of the reviewed works used K-NN or any of its variants for classification involving stance detection. We evaluate the system using the benchmark dataset, SemEval-2016 task 6 [4]. This is a set of tweets with defined targets and stance and is a standard dataset used for evaluating stance detection techniques. We also use t-test to confirm the statistical significance of our assessment. Finally, we compare our system's performance against recent state-of-the-art works on detecting stance using the same dataset.

The rest of this paper is divided as follows. In Section 2, we examine some related works focusing on feature extraction and classification methods for stance detection. Section 3 covers the SemEval-2016 task 6 dataset which we use for evaluation purposes. Our proposed system for stance detection is presented in Section 4. In Section 5 we present and analyze the results. Finally, we conclude with possible future extensions in Section 6.

2. Related work

There is a growing interest in performing stance detection on microblogs [4, 18]. For all the works that we reviewed, the end classification was confined to three choices: "favor", "against", and "neither". Since the purpose of the stance detection techniques, in general, was to determine the wide feelings that social media users have toward certain controversial and/or specific topics, such as the candidates in a presidential election. In all, the underlying objective was to determine if the social media users exhibited strong feelings towards the subjects of their posts or tweets and if those feelings were in favor or against it. For reporting the performance, we use the macro-averaged F-score (F-score for short, in this Section), the standard metric used for the stance detection task.

In the reviewed works, the datasets which the authors examined were a large collection of individual tweets, or Facebook posts gathered over a specific period. There is an increase in competitions of stance detection for different languages (e.g., SemEval-2016 task 6 [4] is for English while IberEval-2017 [18] is for Spanish and Catalan). In the majority of instances, the authors looked specifically for tweets or posts that made mention of *hot button* political, or social issues. For instance, [9] looked for tweets (using the SemEval-2016 task 6 dataset, or just SemEval-2016 dataset in this Section) that contained the following words and phrases: "atheism", "climate change" etc. On the other hand, [19] examined thousands of documents that contained the terms "marijuana", "Obama", etc as part of detecting stance in H&N14 dataset [20]. The IberEval-2017 dataset tackles a single target, the "independence of Catalonia", where the collected tweets were in Spanish or Catalan. Here, our focus is reviewing works on stance detection for English tweets.

Early forms of tweet classification had a human aspect with people classifying the tweets to gather opinions of other people on various subjects. However, the number of tweets that people have to go through is huge, making the mining process a gigantic ordeal [21, 22]. In this regard, automation of the process is largely taking on the important subject with various studies being developed to address the issue. A recent study, for instance, developed a semi-supervised approach that was intended to handle the tweet-classification where human labeling of stance was not required. Instead, high precision hashtags that were stance-bearing were used instead of relying on human-stance classification processes [23]. Using the SemEval-2016 dataset, the best-reported performance was F-score of 53.6%.

We will focus on features and how they are extracted for the stance detection task. In organizing the related works, we decided to divide the review into two parts. The first part is for works that handcraft-based their feature extraction techniques, and the second part is for those work that used deep learning (DL) for detecting features.

2.1. Feature extraction (handcraft-based)

We will start by reviewing works on stance detection that handcraft-based their feature extraction. Mourad et al. [24] examined the use of a majority vote classifier of three classifiers: SVM, random forest (RF), and Gaussian Naïve Bayes for stance detection in tweets where they used Relief as feature extraction on SemEval-2016 dataset. They reported an overall F-score of 70.04%.

Ebrahimi et al. [25] proposed a probabilistic approach for stance detection. They used linear SVM trained on word n-grams ($n = 1$ to 3), and character n-grams ($n = 2$ to 5). The authors presented a log-linear model where stance, the target of stance, and the sentiment (STS) of the tweet were combined, allowing for a multi-way interaction.

They aimed to discriminate sentiment features from target features for classifying the stance. Testing on SemEval-2016 dataset, they reported F-score = 71.03%.

Liu et al. [26] presented a supervised approach to the stance detection field. Their system, based on the RF model, uses gradient decision trees to merge all included classifiers into an ensemble system, which is extremely effective at retrieving minority classes. While this system is effective at classifying all kinds of stances, it has its challenges in the form of analysis of political tweets that were largely sentimental. Additionally, such tweets were dependent on the subject in question with the same words possibly having different meanings depending on the stance in question. The high number of irrelevant features—that were an inherent feature in this model—required that features were selected predominantly to further determine the stance of any tweet with possible ambiguity issues. The reported F-score for the system was 63.6%.

While sentiment analysis is a critical aspect of the stance detection models and processes, it is not the most or the only critical feature. Using SVM classifier and different features sets, [27] reported their best performance of F-score of only 59.21% on the SemEval-2016 dataset. The authors noted that while the n-grams and word embeddings alone provided for the highest micro F-score of 70.32% (and macro F-score of 59.08%), the sentiment lexicon features are beneficial if one is interested in a higher macro F-score of 59.21%, though micro F-score dropped slightly to 69.84%. Other models evaluated tweets at both, the character level and the word level to determine stance.

Dias and Becker [28], on the other hand, recognized that negative or positive connotations can accompany an analysis of a tweet without the tweet having the same negative or positive connotation on the stance itself. In this regard, their stance prediction model is effective in the sentiment analysis sphere. This model used a variety of stance systems to classify data that included opinion targets, and the most common phrases that may have been used to connote the stance itself via weakly supervised learning procedures. Their reported F-score was 55.36%. Other models were developed that can generalize their prediction on any target group based on their learning criterion. One such system [29], where they reported an F-score of 32.7% on their testing data.

Various approaches have been used for stance detection exercises. Bøhler et al. [30] created a system that uses word vectors and utilized an unsupervised learning algorithm. The system combined the use of shallow features as well as GloVe vectors for word representation in tweets. The system uses various classifiers and combines them into some sort of majority voting classifier that dramatically improved on the baseline method, one that only made use of a single approach. They achieved an F-score of 62.47%.

Elfardy and Diab [16] proposed an SVM based supervised system that uses lexical, sentiment, semantic and latent, and frame semantic features to identify the stance of a tweeter for specific targets using SemEval-2016 dataset. In the competition, their reported F-score was 63.6%. Zhang and Lan [31] devised a two-step learning model for stance detection by addressing relevance as well as the orientation of the tweets. Their list of features can be grouped into the categories: traditional linguistic (e.g. n-grams), similarity (e.g. JSD similarity), topic (e.g. Sent2Topic), sentiment lexicon, tweet specific, and word vector features. The aim is to determine if the tweet is relevant to the given target and whether it supports the given target. To test their system on the SemEval-2016 dataset, given the diversity of the targets, the authors built a unique model for each target. Their reported F-score was 63.34%.

Zotova [32] proposed a system based on *tf-idf* and SVM, the feature vector is generated by applying lemmatization. Each word in the tweet is lemmatized; where a word is removed if it did not have a root. Each tweet's word list is tokenized to map it to *tf-idf* scoring to generate the feature vector. The *tf-idf* scoring is based on the unigram's dictionary built from the training dataset. The proposed system was evaluated on the SemEval-2016 dataset, where the reported F-score was 56%.

Dey et al. [14] used an SVM based two-phase approach for stance detection in tweets. In the first phase, they determine if the tweet contains an opinion, and if so, it will be passed on to the second phase, where the model decides if the tweet is either positive or negative regarding the given target. This is a heavy duty system as the authors relied on MPQA (or Multi-Perspective Question Answering) Opinion Corpus, which is a collection of documents that are manually annotated for opinions and other private states (i.e.,beliefs, emotions, sentiments, speculations, etc) [33].[1] For phase one they used weighted MPQA subjectivity-polarity classification and WordNet-based potential adjective recognizer. And for phase two they used SentiWordNet and MPQA-based sentiment classification, frame semantics, character level n-grams etc. This system achieved an F-score of 74.44% on the SemEval-2016 dataset.

In another study, [34] developed SVM based system that learns in two steps; in this system, the first step is to determine whether the tweets are subjective or objective to the stance. And in the second step, the tweets are classified as either negative, positive, or neutral. There are various classes of models that were used in developing these stance detection models such as unigram and feature-based models. The reported F-score for unigram and feature-based models were 72.4% and 68% respectively. In [35] proposed a tree kernel-based model that uses tree representations for the classification of the tweets. Their best performance F-score was 60.83% using their kernel and senti-features model.

Lai et al. [36] presented feature vector that consists of the following: bag of hashtags, bag of hashtags plus, bag

[1] Available at, `mpqa.cs.pitt.edu`.

of mentions, and bag of mentions plus. In addition of two more features for replied tweets: bag of hashtags for replies, and bag of mentions for replies. The authors objective was to investigate user level stance on the 2016 referendum to reform the Italian Constitution. The stance at user level differs from the stance at tweet level. In the latter, we infer the stance of a single anonymous text, where in the former we try to infer the stance from multiple texts written by the same user. For this, they defined a triplet as a set of three tweets (a tweet, a retweet, and a reply) all written by the same user in a 72-hour window. Their corpus, ConRef-STANCE-ita is made of 968 triplets which was manually annotated by two independent native speakers. For classification they used linear SVM. Following five-fold cross-validation, they reported their best performance of F-score = 76% when using the three hashtags: bag of hashtags plus, bag of mentions plus, and bag of hashtags for replies. In a follow up work, Lai et al. [37] proposed MultiTACOS. This system classified the stances in tweets using four feature groups: stylistic, structural, affective, and contextual. Each group had its own set of text representations. For example, for stylistic features (e.g., bag of words, bag of part-of-speech), structural features (e.g., bag of Twitter marks, bag of hashtags), affective features (sentiment-related and emotion-related), and for contextual features (e.g., domain knowledge, user community knowledge). Their system was evaluated using SVM, Naïve Bayes, and logistic regression (LR) on two targets ("Hillary Clinton", and "Donald Trump") from the SemEval-2016 dataset. Their highest reported results were 64.51% using SVM for the target "Hillary Clinton", and 55.74% using LR for "Donald Trump".

Yan et al. [38] proposed a framework for stance detection on their dataset that is related to the 2016 US presidential election. The dataset consists a total of 3240 tweets collected from 310 Clinton supporters, 412 supporters of Trump. They generated a feature vector with 128 dimensions for selected hashtags used by the Clinton and Trump supporters. For classification, the authors used RF and reported a performance of F-score = 81.56%.

Lai et al. [39] proposed a supervised system for detecting stance towards Hillary Clinton and Donald Trump using the SemEval-2016 dataset. The system is based on domain knowledge which they assumed will improve the performance of stance detection. The proposed features are part of three groups: sentiment, structural, and context-based. For sentiment features, they used four lexica information, including LIWC (Linguistic Inquiry and Word Counts), and DAL (Dictionary of Affect in Language). Structural features consist of three parts: frequency of hashtags, screen names and punctuation marks information. Lastly, context-based features consist of six parts: targetByName, targetByPronoun, targetParty, targetPartyColleagues, targetsOppositors, and nobody (tweets that are difficult to infer their stance). Gaussian Naïve Bayes classifier was trained on both targets, the highest reported results were 71.21% for "Hillary Clinton", and 68.29% for "Donald Trump".

2.2. Feature extraction (DL based)

Next, we examine researches that used deep learning techniques. Sun et al. [19] presented a hierarchical attention network (HAN) to weight the importance of different linguistic information and learn the mutual attention between the linguistic information and the document. The authors tested their system on the SemEval-2016 dataset and reported the performance of an F-score of 61%. They also applied their system on the H&N14 dataset [20], where they reported slightly better performance, an F-score of 62.18%.

Siddiqua et al. [10] proposed a neural ensemble model (PNEM) which uses multi-kernel convolution filters to extract higher-level feature tweet then the feature sequences are become an input to Bi-LSTM (bi-directional long-short-term-memory) and nested LSTMs to learn long-term dependencies. They evaluated their system on the SemEval-2016 dataset where their reported F-score was 72.11%.

Benton and Dredze [40], on the other hand, proposed a semi-supervised pre-training method to predict user embedding using recurrent neural network (RNN) where they evaluated their model on SemEval-2016 dataset and scored 53%. Wei et al. [41] developed a neural network machine system for stance detection. During the development stage, the system learns words from the Google News database. Expressions are separated into sets and subsets with different models within the system meant to ensure accuracy. Results from the sets were evaluated in the second phase of sub-set classification to determine the polarity of a tweet based on a manually annotated polarity mechanism. Their reported F-score was 67.3%.

Dey et al. [15] used a two-step detection system for tweet stance detection. For the first stage, they used a combination of LSTM and attention embedding for subjectivity analysis of the tweets, and for the second stage, they used the same LSTM but for sentiment analysis on the subjective tweets. Their model was evaluated on SemEval-2016. The reported performance F-score was 68.84%.

Vijayaraghavan et al. [42], proposed a system for stance detection where they used both character and word level convolutional neural networks (CNN), a neural network approach. The input to CNN was limited to 140 characters, where tweets are encoded using a one-hot vector, hence, a tweet is represented as a binary matrix. While it is an effective method, analyzing at the character level is extremely difficult and requires larger datasets. Their proposed system F-score was 63.5%.

A comparison of the neural networks-based approach and the feature-based approaches in the field of stance detection shows that feature-based approaches are more effective at testing data provided. However, CNN was more effective at the cross-validation of the provided data [12, 43], where the reported F-score respectively were 61.8% and 68.79%.

Schiller et al. [44] presented a multi-dataset learning (MDL) using BERT architecture for its feature vector. BERT (Bidirectional Encoder Representations from Transformers) is an open-sourced NLP pre-training model developed at Google [45]. Their MDL model used ten datasets from five different domains. The model F-score was 71.62% when tested on the SemEval-2016 dataset.

Hanawa et al. [46] proposed a novel neural network model that can encode the given text using bi-directional LSTM using a dataset generated from Wikipedia. Their model was evaluated on their novel dataset for stance. The dataset consists of 6701 tweets covering seven topics linked to Wikipedia articles. Their best F-score was 50.7%. Though the score is relatively low, it is higher than the score obtained by the baseline method from the state-of-art when applied on the same dataset.

3. Dataset used

For this work we used the benchmark training and testing dataset provided by SemEval-2016 task 6 [4].[2] Under this dataset there are two tasks, designated as task A and task B. In this work we will be working on task A. This dataset consists of tweets of the following five target subjects: "atheism", "climate change is a real concern", "feminist movement", "Hillary Clinton", and "legalization of abortion". Why did the organizers choose such random subjects? Primarily, because political and social topics are the easiest to use for assessing the stance detection techniques [15]. The individuals who tweet or post about these issues tend to have very strong feelings either for or against them.

The annotated data consists of a target topic, a related tweet, and the stance of the tweet towards the target. For a sample of the training set, see Table 1. The dataset consists of 4,163 tweets split into two disjoint sets: 2,914 tweets for training, and 1,249 tweets for testing. The stance label is either one of the three: "favor", "against", and "neither". The authors crowdsourced the task of annotating the stance, where each tweet was annotated by eight respondents [47]. Tables 2-3 summarizes the dataset statistics. Overall, the dataset is small. This makes the stance classification task even more challenging. The training set is less than 3000 tweets (ranging from a minimum of 395 to a maximum of 689 tweets per topic). Also, we have a class imbalance in the samples per topic. For the topic "climate change", less than 4% of the training samples are classified as "against".

4. Proposed approach

For stance detection we propose the following three-phase process: (a) tweets preprocessing; here we clean and normalize tweets to generate word and stem lists, (b) features generation; in this step, we create two dictionaries (for stems and normal words), and sentiment attribute to be used for generating a fused features vector, and lastly (c) classification; all the instances of the features are classified based on the list of topics. We will go through each of the phases in detail.

4.1. Preprocessing the tweets

The tweets require some cleaning and normalization before features are extracted. There are digits, codes, and some special characters, e.g. emojis. All these will be dealt with in the preprocessing stage, where we: (i) remove any character, not in the alphabet range of the English language; (ii) normalize the sequence of repeated characters, such as in "Helloooo", which we call speech effects. These are common in social media posts and are mainly intended to emphasize. Untreated speech effect will cause the different variants to be treated as different words and these will likely be ignored by *tf-idf* as being a rare word. We will normalize the variants to a single form, so "Helloo", "Helloooo", etc will all be mapped to "Hello". This normalization was used to preprocess tweets in [17, 35]; (iii) remove stop-words and apply spell correction on all words using the approach in [48], and finally (iv) generate both words and stems lists by tokenizing all the words in the tweet from which we can generate a list of stems. Algorithm 1 lists the preprocessing phase.

4.2. Generating fused dictionary

Algorithm 2 lists the steps for generating the dictionaries. Going through each tweet in the training dataset, we compile the full list of words and stems. This is followed by the process of generating a unique list of words and stems, which we call dictionaries D_w and D_s, respectively. Each entry in the dictionary holds the entry (word or stem) along with its frequency. The purpose of the dictionary is to capture the frequency of unique words and stems in the training dataset. These dictionaries will be used to compute the *tf-idf* scores across the dataset (see Algorithm 3), which is part of the feature vector.

4.3. Generating feature vector

Our feature vector θ consists of up to three different categories. In [17, 49], the authors used top-k words as their feature vector. In this work, we use three merged lists instead. The feature vector $\theta(k, \mathit{flag\_si})$ for the tweet t consists of: the top-k words list (SV_w), the top-k stems list (SV_s), and the sentiment attribute. The first two are list of *tf-idf* scores, arranged in descending order (Algorithm 3 line 4). The constant k, which controls their size, is input to Algorithm 3. The sentiment attribute is used to capture the sentiment in the tweet. Its optional inclusion, as part of the feature vector, is controlled by the binary flag *flag_si*.

[2] The dataset is available at `http://alt.qcri.org/semeval2016/task6/`

Table 2: Distribution statistics about SemEval-2016 task 6 dataset (stance). Ref: [4]

		Training				Testing			
		Stance (%)				Stance (%)			
Target	# total	Favor	Against	Neither	Count	Favor	Against	Neither	Count
Atheism	733	17.93	59.26	22.81	513	14.55	72.73	12.73	220
Climate change	564	53.67	3.80	42.53	395	72.78	6.51	20.71	169
Feminist	949	31.63	49.40	18.98	664	20.35	64.21	15.44	285
Hillary Clinton	984	17.13	57.04	25.83	689	15.25	58.31	26.44	295
Abortion	933	18.53	54.36	27.11	653	16.43	67.50	16.07	280
Total	4163	25.84	47.87	26.29	2914	24.34	57.25	18.41	1249

Table 3: Distribution statistics about SemEval-2016 task 6 dataset (sentiment)

		Training				Testing			
		Sentiment (%)				Sentiment (%)			
Target	# total	Positive	Negative	Neutral	Count	Positive	Negative	Neutral	Count
Atheism	733	60.43	35.09	4.48	513	59.09	35.45	5.45	220
Climate change	564	31.65	49.62	18.73	395	29.59	51.48	18.93	169
Feminist	949	17.92	77.26	4.82	664	19.30	76.14	4.56	285
Hillary Clinton	984	32.08	64.01	3.92	689	25.76	70.17	4.07	295
Abortion	933	28.79	66.16	5.05	653	20.36	72.14	7.50	280
Total	4163	33.05	60.47	6.49	2914	29.46	63.33	7.21	1249

The *tf-idf* [50] is a common method used to calculate the weight of a word in a document. The main idea of *tf-idf* is, if the term frequency of a word in a document is high and this word rarely appears in other documents, then we believe that the word has a good ability in distinguishing categories. The SV_w is the k highest-scoring list of words, a subset of the words from the tweet. This list represents the *tf-idf* scoring of each word w in the tweet ($\in W$, the list of all words from the preprocessed tweet) showing the importance of that word in the documents, tweets in our case. For instance, $\textit{tf-idf}(w_1, W, D_w) = tf(w_1, W) \cdot idf(w_1, D_w)$, where *tf* is term frequency, and *idf* is the inverse document frequency. Similarly, the SV_s is the k highest-scoring list of stems. It is used to adjust the values of the data to capture words with high occurrences used in each topic. For example, the forms "Abortion" and "Abortions" are derived from the stem "Abort". In case a word does not have a stem, then it is excluded from the stems list. Both works [14, 17] used stems as feature vector which did boost their reported accuracy.

The third category in our feature vector is the sentiment information. For this, we rely on NRC (National Research Council Canada) word-emotion association lexicon (or, EmoLex for short) [51]. It is a list of approximately 14,000 English words and their associations with eight basic emotions (anger, fear, anticipation, trust, surprise, sadness, joy, and disgust), and two sentiments (negative or positive). The sentiment attribute is generated by passing the words list to the EmoLex method (Algorithm 3 line 5) which generates emotions and the sentiment association scores for specific text.

4.4. Classification

To evaluate the feature set we use SVM and K nearest neighbor (K-NN).[3] The K-NNs simplicity and relatively high convergence speed make it a popular choice. Though in some applications, it may not produce adequate results (e.g. [52]), however, the fact it has only one parameter (K, the number of neighbors), makes it easy to fine-tune to a variety of situations. For large data sets, the computational demands for classification using the classic K-NN can be prohibitive. To overcome some of the inherent problems with K-NN, over the years researches proposed different extensions of the K-NN. Some of the extensions are, e.g. discernibility-based K-NN (DKNN) [53], weighted K-NN (WKNN) [54], class-based K-NN (CKNN) [53], and exemplar-based K-NN (EKNN) [55]. Unlike the parameter K of the classic K-NN which is pre-selected by the user,

[3] We are following the notation used in the literature. The K in K-NN is not related to the parameter k in the feature vector $\theta(k, \textit{flag\_si})$.

the extensions do not require such and it is determined automatically. Some of these variants were employed successfully for other classification tasks. For instance, in the NLP subfield of automatic text categorization, [56] used DKNN and reported an improved performance over other classifiers including K-NN. We believe it is reasonable we should investigate K-NN and the aforementioned extensions for stance detection. Algorithm 3 covers the feature generation and the classification phases.

For problems with overlapping classes, K-NN may miss some entities of the test set. DKNN appears to provide an advantage over the standard K-NN, by considering the structural properties of the neighbors (which are reflected in a specific score), as well as their distances from each test sample [53]. WKNN assigns weights to the features according to their properties of the neighbors to select the most significant ones, particularly when the number of features is too large. CKNN was introduced to tackle the problem of unbalanced data [50]. It does not take into account several neighbors around a given test sample but instead considers several neighbors from each class. While EKNN classifies a sample based on the mean centroid of those training samples for a given class.

Algorithm 1: Preprocessing the raw tweets.

Input: Raw tweet t
Output: Set of words (W), and stems (S) of t

begin
  Filter out each word w in tweet t where only English alphabets are kept (other characters like, e.g. !, @, #, $, %, etc. are removed)
  Normalize speech effects in t (e.g., "Helloo", "Helloooo", etc are mapped to "Hello")
  Remove all stop-words from t
  Spell correction of each word in t
  Tokenize t to generate the words list $W = [w_1, w_2, \ldots]$
  Generate stems list S from W, where $S = [stem(w_1), stem(w_2), \ldots]$
end

5. Evaluation and results

In this Section we will evaluate our proposed system using the benchmark dataset. For convenience we will divide this Section as follows: (a) evaluation metrics, we briefly delve into the measures used for the evaluation; (b) fine tuning the parameters, were we setup the parameters for the experiments (e.g., cost C for SVM, K of K-NN, etc); (c) experimental results, we report the result of testing under different setting and different classifiers; and (d) discussion, we discuss the result of the experiments and how does it compare to the state-of-the-art.

Algorithm 2: Generating the fused dictionary from the training dataset.

Input: Training dataset d
Output: D_w (D_s) dictionary of unique words (stems) and their frequency

begin
  Initialize lists X and Y
  Update X (and Y) with list of all words (and stems) from each tweet $t \in d$ `// Algorithm 1`
  Generate sorted dictionaries D_w and D_s based on the frequency of unique entries in X and Y
end

5.1. Evaluation metrics

The performance is evaluated using the training and testing set of SemEval-2016 task 6 (Tables 2 and 3), which consists of 4163 tweets, split between five targets where each tweet is labeled based on the stance ("favor", "against", and "neither"). We report the performance in terms of F-score. This is the official metric approved by the organizers of the stance detection task. It is also a common metric that is widely used for sentiment analysis as well. All the systems were evaluated using the macro-average of two labels only. That is, the mean of F-score for the two main classes "favor" and "against" [4]. Let F_{avg} denote this evaluation metric, and is given by,

$$F_{\text{avg}} = \frac{F_{\text{favor}} + F_{\text{against}}}{2}, \tag{1}$$

where F_{favor}, and F_{against} are calculated by,

$$F_{\text{favor}} = \frac{2P_{\text{favor}}\, R_{\text{favor}}}{P_{\text{favor}} + R_{\text{favor}}}, \tag{2}$$

$$F_{\text{against}} = \frac{2P_{\text{against}}\, R_{\text{against}}}{P_{\text{against}} + R_{\text{against}}}, \tag{3}$$

where P_{favor}, P_{against}, R_{favor}, and R_{against} are the precision, and recall for the classes "favor" and "against" (respectively). The F_{avg} can be reported for overall, or for each target separately. We will refer to the F_{avg} calculated for each target as micro averaged, or F_{avg} for short. On the other hand, the F_{avg} for overall is generated by averaging the calculated F_{avg} for each target separately as macro averaged, $F_{\text{overall\_avg}}$ for short.

The F-score was used as the choice metric in the SemEval-2016 competition for evaluating different solutions in stance detection. Note that systems are performing relatively better on the more frequent target classes, and will obtain higher F-scores. On the other hand, to obtain a high F-score, a system has to perform well on all target classes. This evaluation measure does not ignore the "neither" class. By taking the average F-score for only the "favor" and "against" classes, we consider "neither" as

Algorithm 3: Proposed system for stance detection. The input parameter k (the k in top-k) sets the size of the ranked list. The flag *flag_si* controls whether to include/exclude the sentiment information in the feature vector. For the normalization function *norm*( ) we use the standard score (or z-score, as commonly referred to).

Input: Dataset d, dictionaries D_w and D_s, size of the ranked list k, and flag *flag_si*

begin
  foreach tweet $t \in d$ **do**
    Get list of words (W) and stems (S) `// Algorithm 1`
    `// Assuming` $W = [w_1, w_2, \ldots]$
    Generate scoring vectors (in descending order) SV_w and SV_s for t using *tf-idf*, where $SV_w = [\textit{tf-idf}(w_1, W, D_w), \textit{tf-idf}(w_2, W, D_w), \ldots, \textit{tf-idf}(w_k, W, D_w)]$, and $SV_s = [\textit{tf-idf}(s_1, S, D_s), \textit{tf-idf}(s_2, S, D_s), \ldots, \textit{tf-idf}(s_k, S, D_s)]$.
    `//` *EmoLex*`(): a method to generate sentiment attributes`
    Generate a fused feature vector $\theta(k, \textit{flag\_si}) = norm(SV_w) \cup norm(SV_s) \cup \begin{cases} \textit{EmoLex}(W), & \text{if } \textit{flag\_si} = 1, \\ \varnothing, & \text{otherwise.} \end{cases}$
    Do classification of $\theta(k, \textit{flag\_si})$
  end
end

a class that is not of focus or *negative* class in Information Retrieval. Misclassification of negative class samples will affect the F-scores of this metric negatively. If we considered all the classes as part of the metric, and if the negative class is very dominant, then simply each sample will be misclassified and the negative class will obtain high F-score.

5.2. Tuning parameters

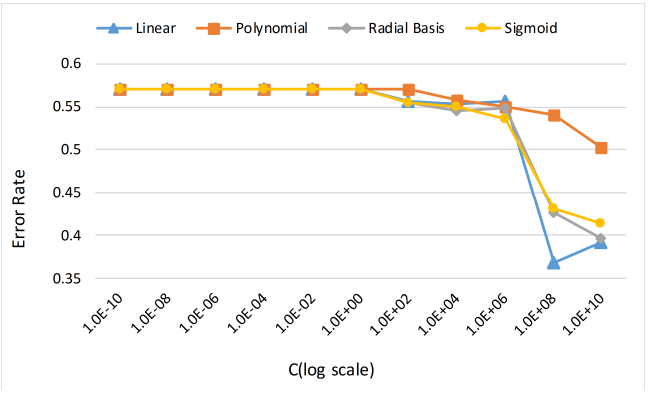

Figure 1: Error rate for different values of cost C for the SVM using different kernel functions.

For experiments there are certain underlying parameters that needs to be tuned. These values will be used in all subsequent experiments in Section 5.3. In the first experiment we want to decide the cost C for the SVM under various kernel functions. For the second experiment, we want to determine the best value of K for the K-NN classifier. The third experiment is to work out the most suitable value of k, this is the parameter that sets the size of the ranked list (which we referred to earlier as top-k). The final experiment investigates the appropriate dimension for principal component analysis (PCA).

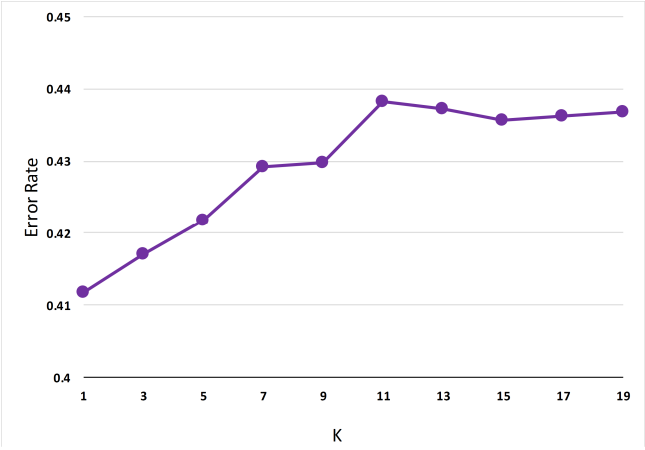

Figure 2: Error rate for selected values of K (number of neighbors) for the K-NN classifier.

When tuning the various parameters, we need to fix the size of the ranked list. We set $k = 10$ (more on that later). This value of k was used when determining C for SVM, the K for K-NN, and the dimension for PCA.

In tuning the parameters (e.g., the cost C for the SVM, etc) we follow the guideline given in SemEval-2016 task 6 [4]. We evaluate our system configurations using five-fold cross-validation on the training data. Typically it goes as follows: four folds are used for the training, and the fifth for testing. This process is repeated five times, randomly picking a different fold for testing, and the other four for training.

For the first experiment, we need a model selection for the penalty factor C in SVM. The parameter C controls the cost of misclassification of the training data in SVM.

Table 4: The F-scores for systems on stance detection task using SemEval-2016 task 6 dataset. The best value in each column is boldfaced.

	Overall			Atheism	Climate	Feminism	Hillary	Abortion
Our system	F_{favor}	F_{against}	$F_{\text{overall\_avg}}$	F_{avg}	F_{avg}	F_{avg}	F_{avg}	F_{avg}
$k = 5$								
WKNN (w/Sentiment)	77.25	64.09	70.67	67.91	71.09	62.54	71.01	66.47
WKNN+PCA (w/Sentiment)	76.55	61.41	68.98	67.02	67.83	62.18	69.94	66.87
WKNN (w/o Sentiment)	76	60.83	68.42	62.36	71.67	61.20	66.86	65.84
WKNN+PCA (w/o Sentiment)	74.30	57.76	66.03	62.17	67.80	59.10	65.36	65.21
$k = 10$								
WKNN (w/Sentiment)	84.05	67.96	76	73.08	**80**	72.77	74.57	75.74
WKNN+PCA (w/Sentiment)	84.49	68.03	76.26	73.43	**80**	72.81	74.67	75.58
WKNN (w/o Sentiment)	84.49	68.36	**76.45**	73.52	79.95	72.99	75.02	75.74
WKNN+PCA (w/o Sentiment)	**84.56**	67.84	76.20	**73.56**	79.95	72.90	75.02	75.78
$k = 15$								
WKNN (w/Sentiment)	78.87	62.39	70.63	66.40	70.86	68.24	70.15	69.35
WKNN+PCA (w/Sentiment)	79.34	62.19	70.77	66.70	71.97	67.53	69.81	69.88
WKNN (w/o Sentiment)	75.37	62.01	68.69	65.48	70.08	62.29	66.30	66.36
WKNN+PCA (w/o Sentiment)	75.26	60.94	68.10	63.52	70.16	61.03	66.00	65.66
State-of-the-art systems (2016–19). Entries ordered on $F_{\text{overall\_avg}}$ score. Unreported values are marked '–'.								
Two-phase SVM [14]	69.53	**79.36**	74.44	72.50	53.59	**78.77**	**79.70**	**83.60**
PNEM [10]	66.56	77.66	72.11	67.73	44.27	66.76	60.28	64.23
Disc-STS [25]	64.43	77.62	71.03	65.52	41.18	57.90	74.48	67.94
Majority vote [24]	–	–	70.04	72.11	46.80	42.56	79.87	62.72
T-PAN [15]	–	–	68.84	61.19	66.27	58.45	57.48	60.21
HAN [19]	–	–	61	70.53	49.56	57.50	61.23	66.16
RNN-HSET [40]	–	–	53	58.20	44.50	51.20	50.90	60.20
Baselines given by the SemEval-2016 task setters (from [4])								
Majority class	52.01	78.44	65.22	42.11	42.12	39.10	36.83	40.30
SVM-unigrams	54.49	72.13	63.31	53.25	38.39	55.65	57.02	60.09
SVM-ngrams	62.98	74.98	68.98	65.19	42.35	57.46	58.63	66.42
SVM-ngrams-comb	54.11	70.01	62.06	53.27	47.76	52.82	56.50	63.71
Participating teams' performances in SemEval-2016 (from [4])								
MITRE	59.32	76.33	67.82	61.47	41.63	62.09	57.67	57.28
pkudblab	61.98	72.67	67.33	63.34	52.69	51.33	64.41	61.09
TakeLab	60.93	72.73	66.83	67.25	41.25	53.01	67.12	61.38
PKULCWM	56.96	74.55	65.76	56.39	40.39	51.32	62.26	61.56
ECNU	60.55	70.54	65.55	61.97	41.32	56.21	57.85	61.25
CU-GWU	54.99	72.21	63.60	55.68	39.41	53.88	51.19	59.38
IUCL-RF	52.61	74.59	63.60	57.93	39.06	51.06	49.84	57.61
DeepStance	58.44	68.65	63.54	52.90	40.40	52.34	55.35	63.32
UWB	57.41	69.42	63.42	57.88	46.90	51.82	59.82	61.98
IDI@NTNU	58.97	65.97	62.47	59.59	54.86	48.59	57.89	54.47
Tohoku	49.25	75.18	62.21	58.90	39.51	52.41	39.81	37.75
Itl.uni-due	48.71	74.75	61.73	52.47	35.50	55.12	44.23	57.25
LitisMind	50.67	72.20	61.44	52.36	39.15	57.16	42.08	45.88
JU_NLP	46.68	74.53	60.60	38.99	42.60	45.65	50.25	41.83
NEUSA	49.03	71.20	60.12	48.90	41.95	52.14	48.53	61.89
NLDS-UCSC	50.90	67.81	59.36	57.19	42.10	48.97	57.27	61.66
WFU/TNT	47.55	70.89	59.22	46.16	42.07	47.91	45.88	45.34
INESC-ID	50.58	64.57	57.58	52.67	44.92	49	50.64	49.93
Thomson Reuters	30.16	62.23	46.19	44.79	35.86	39.37	34.98	38.89

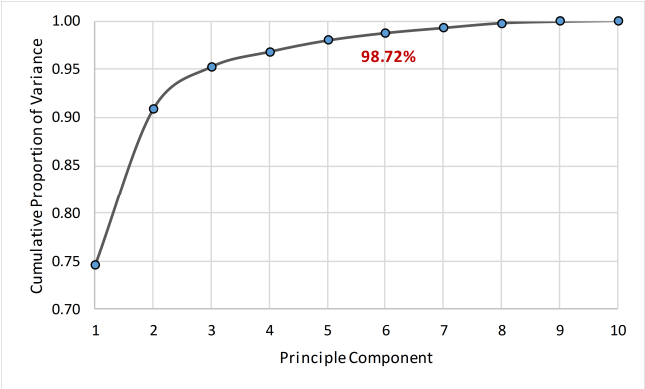

Figure 3: Cumulative proportion of variance for the dataset.

To determine C we will use the feature vector without the sentiment information (see Section 4.3), and with no dimension reduction. After plotting the error rate, we pick C using two factors: the best F_{avg}, and the best performance. Figure 1 plots the error rate for different values of cost C (ranging between 10^{-10} and 10^{10}), and using different kernel functions (linear, polynomial, radial basis, and sigmoid). Among the different kernels, the linear kernel relented the least error rate of 0.37 at $C = 10^8$. Thus, for SVM we will use the linear model with cost $C = 10^8$ which yielded the best F_{avg}. We avoided larger C, since it takes longer to train and validate.

The next tune up is to determine the best K (the number of neighbors) for the classic K-NN classifier. Again, as with the previous experiment, we will use the feature vector without the sentiment information and no dimension reduction, while using $k = 10$. Figure 2 plots the error rate for values of K that are positive odd integers less than 20. We note that the error rate is the least when $K = 1$. Thus, all future references to K-NN, will be denoted 1-NN as it warrants the best F_{avg}.

The third experiment is to determine the most suitable value for k, the size of the ranked list. We have two ranked lists, and they are part of the feature vector. So, certainly, the value of k will impact the classification process. We experiment with the values $k \in \{5, 10, 15\}$, with and without the sentiment information in the feature vector, both of which are inputs to Algorithm 3. This range for k is based on the number of words per tweet, which on average is around 12±2 words. Table 5 lists the overall F_{avg} for three different values of k using the different classifiers. Since the best performance of 76.45% was reported for $k = 10$, this will be our choice for k. However, we will leave the details for later discussion (see Section 5.3). Interestingly, this value of k agrees with the recommendation in [17], where the authors reported that selecting a too small (or too large) value for k impacts the performance. When k is smaller than the average number of words in a document, then we do not capture all the required information, and when it is larger than the average, there will be more padding of the feature vector, thereby losing its focus in W and S lists (the words and stems list in Algorithm 3). It is for this argument why we picked $k = 10$ (close to average number of words per tweet) when selecting a model, e.g. for the cost of C in SVM.

Table 5: Evaluation result using different classifiers, with and without sentiment information for three different values of k (size of the ranked list). The best performance for each k is boldfaced.

		$F_{\text{overall\_avg}}$		
	Sentiment	$k = 5$	$k = 10$	$k = 15$
1-NN	Yes	70.24	61.81	59.85
1-NN	No	66.91	58.65	67.51
CKNN	Yes	66.94	71.57	69.55
CKNN	No	65.54	64.12	64.89
DKNN	Yes	69.11	54.96	58.02
DKNN	No	69.11	56.15	66.27
WKNN	Yes	**70.67**	76.00	**70.63**
WKNN	No	68.42	**76.45**	68.69
EKNN	Yes	67.15	56.65	60.48
EKNN	No	65.60	53.27	64.78
SVM	Yes	58.95	55.37	67.31
SVM	No	63.61	63.13	62.72

The PCA is a statistical procedure used for reducing the dimensionality of large datasets in such a way that most of the information in the data is preserved [57]. We evaluate the use of PCA by selecting the dimension size based on the percentage of cumulative variance, see Figure 3. When the number of principal components is six the cumulative variance reaches 98.7% after which the curve flattens out. We therefore transform the feature vector dimension from ten to six using PCA.

5.3. Experimental results

We evaluate the system using the dataset distribution proposed in SemEval-2016 task 6 (see Table 2). Table 5 lists the overall F_{avg} for each of the tested configurations. We note quite a difference in the performances, which ranged between 53.27% and 76.45%. EKNN with $k = 10$ and excluding the sentiment information was the worst-performing classifier, while the WKNN classifier with a ranked list of size 10 and no sentiment information yielded the best performance of 76.42%. We did not find a fixed pattern indicating when the inclusion of the sentiment information can improve the performance. For example, for the 1-NN classifier, the inclusion of the sentiment did improve the performance for cases $k = 5$ and 10 (a respective improvement of 3.33 and 3.16 in the value of $F_{\text{overall\_avg}}$), but a drop of 7.66 for $k = 15$. While for SVM it was the other way around. Using the sentiment information degraded the performance of SVM for cases $k = 5$ and 10, while improving it when $k = 15$.

Table 6: Evaluation result using different classifiers with and without PCA. Each experiment is repeated twice, once with sentiment information (marked "w/Sent"), and another without the sentiment information (marked "w/o Sent"). We boldfaced the best value in each column.

	F_{avg}											
	Overall		Atheism		Climate		Feminism		Hillary		Abortion	
	w/Sent	w/o Sent	w/Sent	w/o Sent	w/Sent	w/o Sent	w/Sent	w/o Sent	w/Sent	w/o Sent	w/Sent	w/o Sent
1-NN	61.81	58.65	54.78	49.37	47.94	43.50	55.11	52.48	57.65	55.08	72.50	69.59
1-NN+PCA	61.49	58.34	54.96	49.17	48.05	43.50	55.11	52.38	57.71	55.29	72.39	69.55
CKNN	71.57	64.12	72.66	47.74	71.31	75.28	54.55	52.58	57.05	50.34	70.06	46.36
CKNN+PCA	75.19	75.50	73.09	72.74	77.80	78.91	70.47	72.39	73.99	73.54	75.47	75.33
DKNN	54.96	56.15	47.71	47.58	43.22	42.92	47.63	45.07	55.03	50.71	64.00	68.72
DKNN+PCA	54.47	55.57	47.71	46.92	43.22	42.46	47.63	44.78	55.35	51.28	63.70	68.59
WKNN	76.00	**76.45**	73.08	73.52	**80.00**	**79.95**	72.77	**72.99**	74.57	**75.02**	**75.74**	75.74
WKNN+PCA	**76.26**	76.20	**73.43**	**73.56**	**80.00**	**79.95**	**72.81**	72.90	**74.67**	**75.02**	75.58	**75.78**
EKNN	56.65	53.27	54.78	49.37	48.08	43.50	55.02	52.10	56.95	54.56	57.88	53.92
EKNN+PCA	62.03	58.82	54.96	49.17	48.05	43.50	55.11	52.46	57.71	55.29	72.46	69.55
SVM	55.37	63.13	55.90	61.05	57.69	69.29	47.35	60.47	53.10	58.98	53.21	61.36
SVM+PCA	60.19	65.26	60.64	63.31	62.72	64.01	57.36	63.07	56.31	64.65	59.69	64.22

Next, we investigate the effect of sentiment information and PCA using different classifiers. Table 6 reports F_{avg} for each configuration. We note the following insights: (a) the F_{avg} for all classifiers ranged between 53.27% to 76.45%, with WKNN and CKNN being the top-performing classifiers; (b) using sentiment information had an inconclusive impact on the value of F_{avg}; in case of CKNN classifier it boasted the F_{avg} by 7.45%, while in some cases the F_{avg} dropped by 7.76% (using SVM classifier); and (c) using PCA for dimension reduction did not always improve the performance. When using PCA, the highest increase in F_{avg} was for the classifiers CKNN (11.38% when no sentiment information is involved), EKNN (5.55%, also without sentiment information), and SVM (4.82% when considering sentiment information). Of all the possible configurations, the winning performance is the WKNN classifier without the sentiment information and no PCA ($F_{\text{avg}} = 76.45\%$), followed by WKNN classifier with sentiment information and PCA ($F_{\text{avg}} = 76.26\%$), a close second.

This is a side experiment where we look at alternate scoring function for the feature vector. Our feature vector uses the *tf-idf* score of the top-k words and stems. We would like to compare the performance when *tf-idf* scoring is replaced with the Bag-of-Words (BoW) scoring. We applied the BoW scoring on both word and stem lists. Using the same configurations that were used with *tf-idf*, Figure 4 shows the performance expressed in terms of F-score for different classifiers. For most classifiers, we achieved a better F-score when using *tf-idf* scoring over that when using BoW. The only exception was EKNN, where BoW yields a slightly better F-score. The difference in F-score ranged from less than 1% to as high as 14.25%, in favor of *tf-idf* over BoW. The greatest two difference was 14.25% and 8.84% for SVM and WKNN classifiers, respectively.

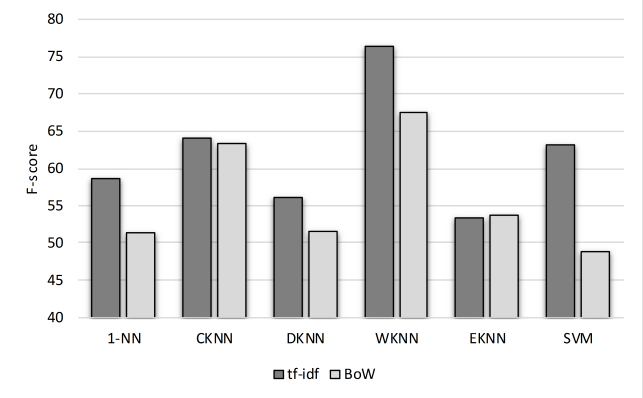

Figure 4: A comparison between *tf-idf* and the Bag-of-Words (BoW) as scoring methods for our feature vector using different classifiers.

5.4. Discussion

From Table 6 we saw that WKNN was the top performing classifier. We will now assess whether the performance of the WKNN classifier statistically significant when compared to others. We will use t-test to confirm our assessment. Table 7 summarizes the results. We observe the following: (a) statistically, there is no significance in the value of F_{avg} when using WKNN classifier with or without sentiment information; (b) there is no statistical significance in F_{avg} when using with or without PCA; (c) statistically, we get significantly better F_{avg} when *tf-idf* scoring is used as opposed to BoW scoring; and (d) WKNN is the best performing classifier.

Table 4 is a comparison between our proposed model and other studies in detecting stance, all using the same

Table 7: Summary of Student's t-test between our proposed configurations using WKNN (weighted K-NN) with different setting, and against other classifiers. The p-values are expressed in scientific E-notation.

WKNN	p-value	Statistically significant?
w/sentiment vs without	9.13E-01	No
w/PCA vs without	9.31E-01	No
Using *tf-idf* vs BoW	3.61E-02	Yes
vs 1-NN	1.04E-05	Yes
vs EKNN	2.37E-08	Yes
vs CKNN	1.58E-04	Yes
vs DKNN	7.18E-07	Yes
vs SVM	1.00E-04	Yes

benchmark dataset, SemEval-2016 task 6. For our system, we report the performance of using the WKNN classifier with all possible configurations (different sizes for the ranked list, with and without sentiment information, and optionally PCA). We also compare against recent state-of-the-art works on stance detection, e.g. [10, 14, 25] etc. Our proposed solution's F-scores were outperforming the best result reported in the state-of-the-art study (i.e. [14]) in both F_{favor} and $F_{\text{overall\_avg}}$ by 14.95% and 2.01% respectively. Moreover, we improved the F-scores for the targets "atheism" and "climate change is a real concern" by 1.06% and 26.36% respectively, over those reported in the same study.

Going over Table 4 we note that all the models including state-of-the-art works (namely, [10, 14, 25]) reported better performance in handling the "against" class, while our model (all configurations) did better in the "favor" class. What is interesting, only 25% of the dataset is in the "favor" class, and 50% in the "against" class (see Table 2). This makes "favor" a minority class. Our worst $F_{\text{favor}} = 74.30$ (WKNN using $k = 5$, with PCA and without sentiment information) beats the best value reported for $F_{\text{favor}} = 69.53$ in [14] among all the competing systems.

Table 8: Measuring the lexical diversity for different tweets groups using type/token ratio (TTR).

Tweets group	TTR
Favour	0.3140
Against	0.2675
Atheism	0.3625
Climate	0.4440
Feminism	0.3411
Hillary	0.3377
Abortion	0.3080

Indeed our model did very well in handling the minority class, but not so well for the majority class using the WKNN classifier. But why so? We investigated the lexical diversity of each class to identify the cause of this behavior. Type/token ratios (TTR) have been extensively used in child language research as an index of lexical diversity [58]. The TTR ratio is computed by dividing the total number of unique words of (types) over the total number of words (tokens). When the value of TTR is high, this indicates a lexical richness of the text. We generated TTR using the preprocessed tweets for each group, and the results are tabulated in Table 8. The ratio for the "favor" class is higher than that of the "against" class (0.314 vs 0.2675). This indicates the lexical richness for the "favor" class. Among all the targets, the highest TTR is for the target "climate" which is 0.444. There is a possible correlation between the increase of accuracy and the lexical richness of the dataset. Looking at the performance on detecting the stance on different targets, we see our model outperforms the state-of-the-art works on targets "atheism" and "climate". Among the five targets, these two are the smallest if we count their number of tweets in the dataset.

6. Conclusions and future work

Social media is part of daily rites of a sizable world population. The focus of this study is to detect the stance of a microblogging social media forum tweets. The objective is to determine if the tweet's author is in favor of a given target, against it, or has a neutral stance towards it. In this study, we proposed a fused feature vector of a ranked list of k topmost occurring words, another for k topmost occurring stems, and the sentiment information. It works well with fairly short documents with limited context such as tweets. We tested our approach using different set of classifiers, which included SVM and different variants of K nearest neighbor (K-NN). These are the 1-NN, discernibility-based K-NN, weighted K-NN, class-based K-NN, and exemplar-based K-NN. When it comes to K-NN, we determined experimentally that using one neighbor (i.e. 1-NN) yielded the best F-score. For each classifier, we experimented with three different values of k (i.e. the size of the ranked list), the option to include (or not) the sentiment information, and the option to apply (or not) the dimension reduction method (PCA). We determined that $k = 10$ is the best choice. The system was evaluated using the SemEval-2016 task 6 benchmark dataset. In reality some of the K-NN variants did better job in detecting stance than the 1-NN, which in turn did better than SVM. Our best performance of macro F-score of 76.45% is when using the WKNN classifier, topping the current state-of-the-art performance by 2.01%. We showed that this indeed is a statistically significant result irrespective of whether the sentiment information was included or not, and if PCA was applied or not. In fact, the WKNN classifier topped the other variants of K-NN and SVM.

For future work, we plan to improve the current techniques for tweets authorship profiling, such as gender [17], age group [59, 60]. Alternative deep learning networks with early exits [61], and ensemble-based approaches [62] will also be explored. We also aim to adapt the system for other languages and applications, e.g. automated Arabic text summarization [63, 64]. Another area of interest is studying aspects of Arabic hadith literature. Hadiths are short narrations originating from the sayings and conduct of Prophet Muhammad. Each hadith consists of two parts, the first of which is the chain of narrators (in reverse chronological order) involved in transmitting the hadith text. A major study in hadith literature is the reliability of the transmission chain, which impacts the grading (or veracity) of the hadith. We plan to develop novel NLP techniques that can aid in profiling the narrators. Narrator profiling may help in testing the credibility of the transmission chain (see [65] for a recent survey of computational and NLP based studies on hadith literature).

Acknowledgments

The authors would like to acknowledge M. Gogate and K. Dashtipour for supporting experiments using convolutional neural networks. The authors would like to extend their thanks to all the anonymous reviewers for their helpful comments.

A. Al-Ghadir and A. Azmi would like to thank the Deanship of Scientific Research at King Saud University for funding and supporting this research through the initiative of DSR Graduate Students Research Support. A. Hussain would like to acknowledge the support of the UK Engineering and Physical Sciences Research Council (EPSRC) Grants Ref. EP/M026981/1, EP/T021063/1, and EP/T024917/1.

References

[1] M. Wojatzki, T. Zesch, ltl.uni-due at semeval-2016 task 6: Stance detection in social media using stacked classifiers, in: Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016), 2016, pp. 428–433.

[2] G. Zarrella, A. Marsh, MITRE at semeval-2016 task 6: Transfer learning for stance detection, arXiv preprint arXiv:1606.03784 (2016).

[3] E. Kouloumpis, T. Wilson, J. Moore, Twitter sentiment analysis: The good the bad and the omg!, in: Fifth International AAAI conference on weblogs and social media, 2011.

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

[5] C. Conforti, M. T. Pilehvar, N. Collier, Modeling the Fake News Challenge as a Cross-Level Stance Detection Task, in: Proceedings of the 27th ACM International Conference on Information and Knowledge Management (CIKM '18), Association for Computing Machinery, 2018, pp. 40:1–40:5.

[6] D. Pomerleau, D. Rao, The fake news challenge: Exploring how artificial intelligence technologies could be leveraged to combat fake news, `http://www.fakenewschallenge.org`, 2017. Accessed: 2020-03-25.

[7] P. Krejzl, B. Hourová, J. Steinberger, Stance detection in online discussions, Computing Research Repository (CoRR) (2017).

[8] G. Gorrell, E. Kochkina, M. Liakata, A. Aker, A. Zubiaga, K. Bontcheva, L. Derczynski, SemEval-2019 task 7: RumourEval, determining rumour veracity and support for rumours, in: Proceedings of the 13th International Workshop on Semantic Evaluation, Association for Computational Linguistics, 2019, pp. 845–854.

[9] B. G. Patra, D. Das, S. Bandyopadhyay, JU_NLP at semeval-2016 task 6: detecting stance in tweets using support vector machines, in: Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016), 2016, pp. 440–444.

[10] U. A. Siddiqua, A. N. Chy, M. Aono, Tweet stance detection using an attention based neural ensemble model, 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), 2019, pp. 1868–1873.

[11] Q. Li, Q. Zhang, L. Si, Tweetsenti: Target-dependent tweet sentiment analysis, in: The World Wide Web Conference, 2019, pp. 3569–3573.

[12] J. Du, R. Xu, Y. He, L. Gui, Stance classification with target-specific neural attention networks, in: Proceedings of the 26th International Joint Conferences on Artificial Intelligence (IJCAI-17), 2017, pp. 3988–3994.

[13] P. Rao, Transfer Learning in NLP for Tweet Stance Classification, `https://bit.ly/3cIKlhW`, 2019. Accessed: 2019-06-22.

[14] K. Dey, R. Shrivastava, S. Kaushik, Twitter stance detection—a subjectivity and sentiment polarity inspired two-phase approach, in: IEEE International Conference on Data Mining Workshops (ICDMW '17), 2017, pp. 365–372.

[15] K. Dey, R. Shrivastava, S. Kaushik, Topical stance detection for twitter: A two-phase LSTM model using attention, in: European Conference on Information Retrieval, 2018, pp. 529–536.

[16] H. Elfardy, M. Diab, CU-GWU perspective at semeval-2016 task 6: Ideological stance detection in informal text, in: Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016), 2016, pp. 434–439.

[17] A. I. Al-Ghadir, A. M. Azmi, A study of Arabic social media users—posting behavior and author's gender prediction, Cognitive Computation 11 (2019) 71–86.

[18] M. Taulé, M. A. Martí, F. M. Rangel, P. Rosso, C. Bosco, V. Patti, et al., Overview of the task on stance and gender detection in tweets on Catalan independence at IberEval 2017, in: 2nd Workshop on Evaluation of Human Language Technologies for Iberian Languages (IberEval 2017), volume 1881, CEUR-WS, 2017, pp. 157–177.

[19] Q. Sun, Z. Wang, Q. Zhu, G. Zhou, Stance detection with hierarchical attention network, in: Proceedings of the 27th International Conference on Computational Linguistics, 2018, pp. 2399–2409.

[20] K. S. Hasan, V. Ng, 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.

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

[22] M. Tutek, I. Sekulic, P. Gombar, I. Paljak, F. Culinovic, F. Boltuzic, M. Karan, D. Alagić, J. Šnajder, Takelab at semeval-2016 task 6: stance classification in tweets using a genetic algorithm based ensemble, in: Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016), 2016, pp. 464–468.

[23] A. Misra, B. Ecker, T. Handleman, N. Hahn, M. Walker, NLDS-UCSC at semeval-2016 task 6: A semi-supervised approach to detecting stance in tweets, in: Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016), 2016, pp. 420–427.

[24] S. S. Mourad, D. M. Shawky, H. A. Fayed, A. H. Badawi, Stance detection in tweets using a majority vote classifier, in: Interna-

tional Conference on Advanced Machine Learning Technologies and Applications, 2018, pp. 375–384.
[25] J. Ebrahimi, D. Dou, D. Lowd, A joint sentiment-target-stance model for stance classification in tweets, in: Proceedings of 26th International Conference on Computational Linguistics (COLING 2016): Technical Papers, 2016, pp. 2656–2665.
[26] C. Liu, W. Li, B. Demarest, Y. Chen, S. Couture, D. Dakota, N. Haduong, N. Kaufman, A. Lamont, M. Pancholi, et al., IUCL at semeval-2016 task 6: An ensemble model for stance detection in twitter, in: Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016), 2016, pp. 394–400.
[27] P. Sobhani, S. Mohammad, S. Kiritchenko, Detecting stance in tweets and analyzing its interaction with sentiment, in: Proceedings of the Fifth Joint Conference on Lexical and Computational Semantics, 2016, pp. 159–169.
[28] M. Dias, K. Becker, Inf-ufrgs opinion-mining at semeval-2016 task 6: Automatic generation of a training corpus for unsupervised identification of stance in tweets, in: Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016), 2016, pp. 378–383.
[29] I. Augenstein, A. Vlachos, K. Bontcheva, Usfd at semeval-2016 task 6: Any-target stance detection on twitter with autoencoders, in: Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016), 2016, pp. 389–393.
[30] H. Bøhler, P. Asla, E. Marsi, R. Sætre, Idi@ntnu at semeval-2016 task 6: Detecting stance in tweets using shallow features and glove vectors for word representation, in: Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016), 2016, pp. 445–450.
[31] Z. Zhang, M. Lan, ECNU at semeval 2016 task 6: relevant or not? supportive or not? a two-step learning system for automatic detecting stance in tweets, in: Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016), 2016, pp. 451–457.
[32] E. Zotova, Automatic stance detection on political discourse in Twitter, Master's thesis, Universidad del Pais Vasco, Spain, 2019.
[33] T. Wilson, J. Wiebe, C. Cardie, MPQA Opinion Corpus, Springer Netherlands, 2017, pp. 813–832.
[34] L. Barbosa, J. Feng, Robust sentiment detection on twitter from biased and noisy data, in: Proceedings of the 23rd international conference on computational linguistics: posters, Association for Computational Linguistics, 2010, pp. 36–44.
[35] A. Agarwal, B. Xie, I. Vovsha, O. Rambow, R. Passonneau, Sentiment analysis of twitter data, in: Proceedings of the Workshop on Language in Social Media (LSM 2011), 2011, pp. 30–38.
[36] M. Lai, M. Tambuscio, V. Patti, G. Ruffo, P. Rosso, Stance polarity in political debates: A diachronic perspective of network homophily and conversations on Twitter, Data & Knowledge Engineering 124 (2019) 101738.
[37] M. Lai, A. T. Cignarella, D. I. H. Farías, C. Bosco, V. Patti, P. Rosso, Multilingual stance detection in social media political debates, Computer Speech & Language (2020) 101075.
[38] Y. Yan, J. Chen, M.-L. Shyu, Efficient large-scale stance detection in tweets, in: Deep Learning and Neural Networks: Concepts, Methodologies, Tools, and Applications, IGI Global, 2020, pp. 667–683.
[39] M. Lai, D. I. H. Farías, V. Patti, P. Rosso, Friends and enemies of Clinton and Trump: Using context for detecting stance in political tweets, in: Mexican International Conference on Artificial Intelligence, 2016, pp. 155–168.
[40] A. Benton, M. Dredze, Using author embeddings to improve tweet stance classification, in: Proceedings of the 2018 EMNLP Workshop W-NUT: The 4th Workshop on Noisy User-generated Text, 2018, pp. 184–194.
[41] W. Wei, X. Zhang, X. Liu, W. Chen, T. Wang, pkudblab at semeval-2016 task 6: A specific convolutional neural network system for effective stance detection, in: Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016), 2016, pp. 384–388.
[42] P. Vijayaraghavan, I. Sysoev, S. Vosoughi, D. Roy, Deepstance at semeval-2016 task 6: Detecting stance in tweets using character and word-level CNNs, arXiv preprint arXiv:1606.05694 (2016).
[43] Y. Igarashi, H. Komatsu, S. Kobayashi, N. Okazaki, K. Inui, Tohoku at semeval-2016 task 6: feature-based model versus convolutional neural network for stance detection, in: Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016), 2016, pp. 401–407.
[44] B. Schiller, J. Daxenberger, I. Gurevych, Stance detection benchmark: How robust is your stance detection?, arXiv preprint arXiv:2001.01565 (2020).
[45] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, 2018. arXiv:1810.04805.
[46] K. Hanawa, A. Sasaki, N. Okazaki, K. Inui, Stance detection attending external knowledge from wikipedia, Journal of Information Processing 27 (2019) 499–506.
[47] S. Mohammad, S. Kiritchenko, P. Sobhani, X. Zhu, C. Cherry, A dataset for detecting stance in tweets, in: Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16), 2016, pp. 3945–3952.
[48] S. Bird, E. Klein, E. Loper, Natural Language Processing with Python, O'Reilly Media, 2009.
[49] W. Liu, D. Ruths, What's in a Name? Using First Names as Features for Gender Inference in Twitter, in: AAAI Spring Symposium: Analyzing Microtext, 2013.
[50] A. Rajaraman, J. D. Ullman, Mining of massive datasets, Cambridge University Press Cambridge, 2012.
[51] P. Sobhani, Stance detection and analysis in social media, Ph.D. thesis, University of Ottawa, 2017.
[52] J. M. Sotoca, J. S. Sánchez, F. Pla, Estimating feature weights for distance-based classification, in: Pattern Recognition in Information Systems (PRIS2003), 2003, pp. 156–166.
[53] Z. Voulgaris, G. D. Magoulas, Extensions of the k-Nearest Neighbor methods for classification problems, in: Proceedings of the 26th International Conference on Artificial Intelligence and Applications (AIA '08), Anaheim, CA, 2008, pp. 23–28.
[54] K. Hechenbichler, K. Schliep, Weighted k-nearest-neighbor techniques and ordinal classification, https://epub.ub.uni-muenchen.de/1769/1/paper_399.pdf, 2004. Accessed: 2020-06-15.
[55] Y. Li, X. Zhang, Improving k nearest neighbor with exemplar generalization for imbalanced classification, in: Proceedings of the 15th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining - Volume Part II, PAKDD '11, 2011, pp. 321–332.
[56] K. Shi, L. Li, H. Liu, J. He, N. Zhang, W. Song, An improved knn text classification algorithm based on density, in: IEEE International Conference on Cloud Computing and Intelligence Systems, IEEE, 2011, pp. 113–117.
[57] I. T. Jolliffe, Principal component analysis, Springer-Verlag, Berlin, New York, 1986.
[58] B. Richards, Type/token ratios: what do they really tell us?, Journal of Child Language 14 (1987) 201–209.
[59] Y. Li, L. Yang, B. Xu, J. Wang, H. Lin, Improving user attribute classification with text and social network attention, Cognitive Computation 11 (2019) 459–468.
[60] L. Burdick, R. Mihalcea, R. L. Boyd, J. W. Pennebaker, Analyzing connections between user attributes, images, and text, Cognitive Computation (2020) 1–20.
[61] S. Scardapane, M. Scarpiniti, E. Baccarelli, A. Uncini, Why should we add early exits to neural networks?, Cognitive Computation 12 (2020) 954–966.
[62] Q.-F. Wang, M. Xu, A. Hussain, Large-scale ensemble model for customer churn prediction in search ads, Cognitive Computation 11 (2019) 262–270.
[63] A. M. Azmi, N. I. Altmami, An abstractive arabic text summarizer with user controlled granularity, Information Processing & Management 54 (2018) 903–921.
[64] B. Elayeb, A. Chouigui, M. Bounhas, O. B. Khiroun, Automatic Arabic Text Summarization Using Analogical Proportions, Cog-

nitive Computation (2020) 1–27.
[65] A. M. Azmi, A. O. Al-Qabbany, A. Hussain, Computational and natural language processing based studies of hadith literature: A survey, Artificial Intelligence Review 52 (2019) 1369–1414.