Disambiguation of features for improving target class detection from social media text

1st Fatima Chiroma
School of Computing
University of Portsmouth
Portsmouth, UK
fatima.chiroma@port.ac.uk

2nd Ella Haig
School of Computing
University of Portsmouth
Portsmouth, UK
ella.haig@port.ac.uk

***Abstract*—The rise of social media has led to an abundance of textual data, as well as the rise of unhealthy behaviours targeted at others (e.g. bullying, hate speech) or at oneself (e.g. suicide). In recent years, machine learning approaches have been employed to detect such behaviours, which tend to constitute a small portion of the social media content and need to be distinguished from other discourse on social media that may discuss such behaviours without displaying that behaviour, e.g. social media posts about helping people who may be at risk of suicide, thus, making this a very challenging task. In the context of machine learning, such behaviours are referred to as target classes, i.e. the main behaviours to be detected. In this paper we proposed an approach for disambiguation of features in relation to their membership to the target class vs. non-target class(es). We validate our approach with a case study on suicide detection and our results show that the proposed disambiguation approach leads to a better detection rate of suicide.**

***Index Terms*—Term disambiguation, Text classification, Suicide-related communications, Social media text**

I. Introduction

The rise of social media has led to many opportunities, such as positive campaigning for charities [5], clean-up after riots [29] and mobilisation in response to natural disasters [27]. Along these positives, however, there are concerns about undesirable behaviours as well, such as cyberbullying [35], hate speech [19] and suicide ideation [7], [8], [22], [32], [33].

While these undesirable behaviours may be in minority (i.e. they are a small percentage from all social media posts), they have a disproportionately negative effect on the people targeted by such behaviours (e.g. [11], [33]. Some of these behaviours, such as hate speech, are considered crimes in some parts of the world, e.g. UK and parts of the EU, and major social media networks (e.g. Facebook, Twitter and YouTube) have internal regulatory policies in relation to hate speech [3].

Given the nature of social media, with continuous vast numbers of new posts, there is a need for automatic detection of such undesirable behaviours, for reasons ranging from conformity to regulations (e.g. remove posts with threatening and abusive content) to providing help and support (e.g. for victims of cyberbullying or persons at risk of suicide). In recent years, there has been an increase in research in this area, with various approaches to automatically detect such behaviours (e.g. [7], [19], [20], [32], [40]). In this paper, we focus on machine learning classification approaches where models are built based on labelled social media data. When building such models, researchers are typically focused on maximising the detection of the behaviour/class of interest; this is referred to as a target class, to distinguish it from other machine learning problems that require models to perform well across two or more classes (e.g. topic classification) [9].

A key factor in text classification is the extraction and selection of features [12], given the fuzzy nature of text, with the same word having different meanings in different contexts [15]. Consequently, research on detection of undesirable behaviours on social media also has looked into this aspect by experimenting with different sets of features (e.g. [7], [8]), employing feature selection methods (e.g. [32]) or developing methods for dealing with ambiguous situations (e.g. [20]).

In this paper, we propose a method for the disambiguation of features from social media text with the aim to improve the detection of target classes. Our method identifies terms that appear in textual instances of the target class, the other class or both[1]. The aim of our method is to minimise the number of terms that are common to both the target class and the other class; this aim is achieved by using the probability of a term appearing in text labelled with the target class vs. text labelled with the other class. Once the number of common terms is minimised, the textual instances are modified by eliminating terms that occur predominantly in the opposite class, i.e. terms that appear in text labelled with the other class are removed from textual instances labelled with the target class and vice-versa.

We validate our approach on the problem of suicide detection using several machine learning algorithms, i.e. Decision Tree (DT), k-nearest neighbour (kNN), Naïve Bayes (NB), Random Forest (RF) and Support Vector Machine (SVM). Moreover, we use our approach in conjunction with other approaches, including using two forms of text representation (bag-of-words and word embedding), four methods of feature selection (chi-square, entropy, low variance and firefly algorithm) and two methods of feature extraction (Latent Dirichlet Allocation and Principle Components Analysis). The experimental results show that when our method is used, the

[1]we focus on binary classification with one target class and one 'other' class; if there are several other classes these are merged into one generic 'other' class

classification performance improves for the target class, as well as overall.

The rest of the paper is structured as in the following. Section II covers related work for the detection of unwanted behaviours from social media text, with a focus on feature disambiguation, including feature extraction and selection approaches. The proposed approach is presented in Section III and its validation on suicide detection in Section IV. The results are discussed in Section V, while Section VI concludes the paper and outlines directions for future work.

II. Related Work

Several studies investigated the detection of cyberbullying (e.g. [21], [40]), hate speech (e.g. [20], [36]) and suicide (e.g. [7], [8], [22], [32], [33]) from social media. In the following paragraphs, we review previous work in relation to feature extraction and selection approaches.

Two broad approaches have been used to represent text in a structured format, i.e. bag-of-words/bag-of-n-grams and word embedding. Word embedding representations are obtained by training deep neural networks and they have the main advantage of reduced dimensionality [17], [25]. These representations are relatively recent in comparison to bag-of-words/bag-of-n-grams representations and have been less explored in the area of unwanted behaviour detection from social media. Examples of studies using word embedding as representation are the detection of cyberbullying (e.g. [1], [4]), hate speech [28], [41] and suicide ideation [10], [30].

The bag-of-words/bag-of-n-grams representations have been extensively used in this area, either on their own (e.g. [8], [16], [36], [37], [40]) or in combination with other features such as lexicon features (e.g. [13], [14], [38]), topic features extracted through approaches such as Latent Dirichlet Allocation (LDA) (e.g. [14], [38]), part-of-speech tags (e.g. [14], [37]), or various combinations of these features (e.g. [7], [14], [31], [34], [37], [38]).

A feature selection approach has been proposed for suicide detection using the firefly algorithm (an evolutionary computation algorithm based on the behaviour of fireflies) and some improvement has been shown in comparison with other machine learning approaches without the use of the feature selection [32].

Other feature selection methods used for text classification of unwanted behaviour are filter-based methods [12] using chi-square and entropy-based metrics (e.g. [2], [26]).

For text classification in general and classification of unwanted behaviours, feature extraction methods such as Latent Dirichlet Allocation (LDA) and Principle Components Analysis (PCA) have been widely used (e.g. [6], [7], [14], [38]).

A fuzzy approach to text classification for hate speech has been used with two training stages, with the second stage dealing with ambiguous instances [20]. Experiments were conducted with both bag-of-words and word embedding representations and the results showed that the fuzzy approach outperformed well-known classifiers such as Decision Tree, Naïve Bayes and SVM, as well as deep neural networks [20].

Conceptually, the closest work to the proposed approach is the second stage of the fuzzy approach proposed by [20], as in both the aim is to reduce ambiguity; while [20] works at instance level, our approach is at terms level.

On the other hand, our approach is also similar to feature selection methods, as it involves selecting the features and attributing them to particular sets, based on which the textual instances are transformed (i.e. terms are removed form these instances). The difference between our proposed method and previous work is mainly in the fact that we modify the textual instances prior to their transformation in a structured format in preparation for machine learning. To the best of our knowledge, none of the previous methods looked at modifying textual instances for the purpose of feature disambiguation.

III. Term Disambiguation

The aim of the proposed method for disambiguation is illustrated in Fig. 1. We are presenting the proposed method for a binary classification problem, where we denote the two classes as the target class and the other class.

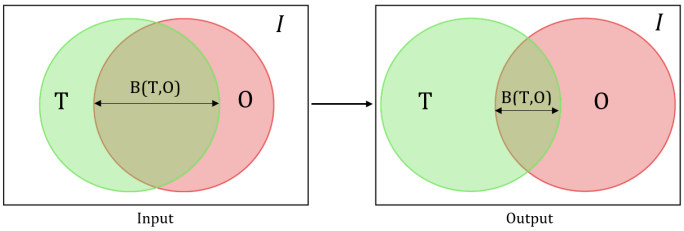


Fig. 1. Disambiguation of terms; I is the set of textual instances; T is the set of terms that occur only in the target class text; O is the set of terms that occur only in the non-target class text (designated as other); $B(T,O)$ represents the set of terms that occurs in both the target class and the other class text. The aim of our proposed method is to increase the size of the T and O sets and minimise the size of the $B(T,O)$ set.

When the same words or terms appear in instances labelled with the target class, as well as instances with the other class, the performance of machine learning algorithms tends to decrease due to the ambiguity of these terms and their unclear relation to the target and other classes. To solve this ambiguity problem we aim to understand which terms are predominantly associated with one of the two classes and to minimise the number of terms that are common to the two classes, as illustrated in Fig. 1.

Algorithm 1 documents the steps performed for achieving the disambiguation aim. We start with the full set of textual instances, i.e. I, which is preprocessed by converting to lowercase and removing punctuation, numbers, URLs and non-ASCII characters. As the validation of the proposed method is done on Twitter data, we also excluded the usernames, # symbol and "rt" terms, which some users place at the beginning of a tweet to show that it is a retweet. After preprocessing, the text is tokenised into unigrams, bigrams and trigrams (see lines 1-8 in Algorithm 1).

The next steps are to identify the unigram terms that appear in instances with the target class label (stored in set T) (lines 9 to 11 in Algorithm 1) and the ones that appear in instances

Algorithm 1: Feature attribution to the T, O and $B(T,O)$ sets

input : Set I of all textual instances; set I_T of target class instances; set I_O of other class instances
output: The sets of terms T, O and $B(T,O)$

1: $T = \emptyset$
2: $O = \emptyset$
3: $B(T,O) = \emptyset$
4: **for** each instance of text in set I **do**
5: convert to lowercase
6: remove punctuation, numbers, URLs, non-ASCII characters, usernames, #, 'rt'
7: tokenise text into *unigrams*, *bigrams* and *trigrams*
8: **end for**
9: **for** each instance of text in set I_T **do**
10: append each *unigram* term t_i to set T
11: **end for**
12: $n = |T|$
13: **for** each instance of text in set I_O **do**
14: append each *unigram* term t_i to set O
15: **end for**
16: $m = |O|$
17: **for** $i = 1$ to n **do**
18: **for** $j = 1$ to m **do**
19: **if** $t_i = t_j$ **then**
20: $T = T - \{t_i\}$
21: $O = O - \{t_i\}$
22: $B(T,O) = B(T,O) \cup \{t_i\}$
23: **end if**
24: **end for**
25: **end for**
26: **for** each term t_i in $B(T,O)$ **do**
27: calculate term frequency in I: $tf(t_i, I)$
28: calculate term frequency in I_T: $tf(t_i, I_T)$
29: calculate term frequency in I_O: $tf(t_i, I_O)$
30: $P(t_i,T) = \dfrac{tf(t_i,I_T)}{tf(t_i,I)}$
31: $P(t_i,O) = \dfrac{tf(t_i,I_O)}{tf(t_i,I)}$
32: $PD(t_i) = |P(t_i,T) - P(t_i,O)|$
33: **if** $PD(t_i) >= 0.5$ **then**
34: **if** $P(t_i,T) > P(t_i,O)$ **then**
35: $T = T \cup \{t_i\}$
36: $B(T,O) = B(T,O) - \{t_i\}$
37: **else**
38: $O = O \cup \{t_i\}$
39: $B(T,O) = B(T,O) - \{t_i\}$
40: **end if**
41: **else**
42: repeat steps 6-34 using *bigram* terms
43: **end if**
44: **if** $B(T,O) \neq \emptyset$ **then**
45: repeat steps 6-34 using *trigram* terms
46: **end if**
47: **end for**
48: Return T, O, $B(T,O)$

Algorithm 2: Terms removal from textual instances

input : Set I_T of target class instances; set I_O of other class instances; the sets of terms T and O
output: Set J of textual instances after term removal

1: $J = \emptyset$
2: **for** each instance I_i of text in set I_T **do**
3: **for** each term t_j in the instance **do**
4: **if** $t_j \in O$ **then**
5: $I_i = I_i - \{t_j\}$
6: **end if**
7: $J = J \cup \{I_i\}$
8: **end for**
9: **end for**
10: **for** each instance I_i of text in set I_O **do**
11: **for** each term t_j in the instance **do**
12: **if** $t_j \in T$ **then**
13: $I_i = I_i - \{t_j\}$
14: **end if**
15: $J = J \cup \{I_i\}$
16: **end for**
17: **end for**
18: Return J

with the other class label (stored in set O) (lines 13 to 15 in Algorithm 1).

In lines 17 to 25, the overlapping terms between the two sets, i.e. T and O, are identified and stored in the set $B(T,O)$ (B stands for both).

The following steps aim to reduce the size of the $B(T,O)$ set by looking at the probabilities of the terms appearing in sets T and O. The probabilities are calculated based on term frequency; for example, the probability of term t_i appearing in set T, i.e. $P(t_i,T)$, is the frequency of term t_i in I_T (i.e. how many times the term appears in instances labelled with the target class) divided by the frequency of term t_i in the whole set of instances I (line 30).

If the probability difference $PD(t_i)$ is equal to or higher than 0.5 (i.e. one of the sets has more than the appearances of the other set plus 50%, meaning that the term appears at least 3 times more in one set than the other) (line 33), the term is assigned to the set with the higher probability and removed from the $B(T,O)$ set (lines 34-39).

If the probability difference $PD(t_i)$ is less than 0.5, meaning that the term does not appear predominantly in one of the sets, the process is repeated by looking at terms from *bigrams* (line 42). If after this process, there are still terms in the $B(T,O)$ set, the process is repeated again with terms from *trigrams* (lines 44-46). The algorithm ends by returning the T, O and $B(T,O)$ sets.

The next step is the removal of terms from textual instances according to the term membership given by Algorithm 1. This process is displayed in Algorithm 2.

If a term from the O set appears in an instance labelled with the target class, the term is removed from the instance (lines 2-9 in Algorithm 2). The same process is applied for

terms of set T appearing in instances labelled with the other class (lines 10-17). The algorithm returns set J of instances with removed terms.

The next section describes how the proposed approach has been applied for the problem of suicide detection. For the validation of the proposed approach, comparison with several feature extraction/selection methods, as well as two text representations are used.

IV. Suicide detection using the proposed approach

In this section we describe the use of the proposed approach on the problem of suicide detection and evaluate it against several other approaches. Fig. 2 illustrates the difference in the process when using our approach vs. typical approaches. In typical approaches, the data is preprocessed and transformed in preparation for classification and evaluation. For our approach, the process flow is illustrated using the dashed line, indicating that the data (i.e. the textual instances) are modified prior to transformation, after which the typical approach is followed.

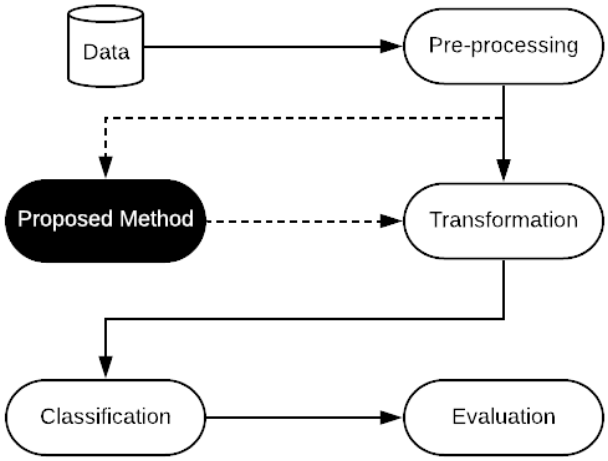


Fig. 2. Experimental Approach

A. Data

The data used in these experiments are short informal texts from Twitter containing topics about suicide that may or may not imply suicidal intent. The data was originally collected and labelled by [6]. They developed a lexicon of terms using posts from Web forums, bogs and microblogs discussing suicidal themes for support and prevention. Each post was annotated to indicate if the author of the post was suicidal or not. The suicidal posts were analysed to identify keywords and phrases suggesting possible suicide intent; 62 such terms were identified which were used to collect data from Twitter using the Twitter Streaming Application Programming Interface (API). Twitter data was also collected using the names of reported suicide cases in England. A sample of around 2000 tweets was extracted and annotated using the CrowdFlower crowdsourcing service[2]. The annotators were asked to assign one of seven categories to each tweet; the seven categories were identified by experts in suicide studies to capture people's general representation when communicating on suicide topics [6]. Only the data with at least 75% annotator agreement were retained. The data distribution is displayed in Table I.

[2]http://www.crowdflower.com; CrowdFlower was rebranded in 2018 as Figure Eight

TABLE I
7-class dataset (adapted from [6])

No	Class Name	Description	Instances
1	Suicide	Possible suicidal intent	159
2	Flippant	Un-serious reference to suicide	133
3	Campaign	Suicide petitions	158
4	Support	support or information	178
5	Memorial	Condolences or memorial	142
6	Reports	Suicide reports excluding bombing	165
7	Other	None of the above	129
	Total		**1064**

TABLE II
Binary dataset

No	Class Name	Instances
1	Suicide	159
2 - 7	Other	905
	Total	**1064**

To apply the term disambiguation described in Section III, we transformed the data into a binary-class distribution as illustrated in Table II.

Algorithm 1 described in Section III was applied on the binary dataset – Table III illustrates the original size of the T, O and $B(T, O)$ sets and their size after the application of Algorithm 1.

TABLE III
Term Count

	Original			After proposed method		
Sets	T	$B(T,O)$	O	T	$B(T,O)$	O
Set size	92	250	2326	120	74	2474
I	**2638**			**2638**		

To evaluate the proposed approach, we employ two text representations (bag-of-words and word embedding), four feature selection methods (chi-square, entropy, low variance and firefly algorithm) and two feature extraction methods (LDA and PCA). For each of these, we compare the baseline, i.e. applying these methods on the original dataset, with the application of these methods on the dataset resulting from the proposed approach.

The data preprocessing that is described in the following subsection was applied to the data prior to the application of the text representation, feature selection and feature extraction methods mentioned above. The subsequent subsection describes the transformation of the data for each of the methods.

B. Preprocessing

The preprocessing included the removal of URLs, non-ASCII characters, punctuation, numbers, usernames, 'rt' and #, as well as case conversion, stemming (to reduce redundancy) and POS (Part of Speech) tagging (using the Penn Treebank tag set [24]). However, standard stopwords were not removed as negation words such as in "not killing" might be useful in distinguishing between the target and the other class.

After preprocessing, the original dataset contained 1063 instances, with 159 for the target class and 904 for the other class. The dataset resulting from the proposed method contained 1062 instances, with 158 instances for the target class and 904 for the other class. The reduction of one instance in the other class for both datasets is due to preprocessing; this instance contained only a url, which was removed as part of the preprocessing. The reduction by one instance of the target class for the dataset resulting from the proposed method was due to the application of the method; this instance contain only one term, which was from the set of terms of the other class and consequently was removed when applying the method.

C. Transformation

Two types of term representations were used, i.e. bag-of-words and word embedding. The employed bag-of-words representation used unigrams and the Inverse Document Frequency (IDF) as the term frequency metric displayed in Equation 1.

$$IDF(t_i) = log\frac{1+n}{1+DF(t_i)} + 1 \quad (1)$$

where t_i represents a term (unigram), n is the total number of instances in the dataset and $DF(t_i)$ is the number of instances that contain the t_i term[3].

The bag-of-words representation led to 2638 features for both the original dataset and the dataset resulting from the proposed method, as both datasets have the same terms (but different IDF values for those terms).

To obtain the word embedding representation, doc2vec [17] was used. For training the doc2vec model, the following parameters were used: a context window size of 5, minimum word frequency of 2, sampling rate of 0.001, learning rate of 0.025, minimum learning rate of 1.0E-4, 200 layers, batch size of 10000 and 40 epochs; Distributed Memory (DM) is used as the sequence learning algorithm, with a 5.0 negative sampling rate. The number of features for both the original dataset and the dataset resulting from the proposed method was 200 (corresponding to the number of layers parameter).

Three established feature selection methods were used with the bag-of-words representation: chi-square, entropy and low variance; details about these methods can be found in [18]. We also applied the firefly algorithm [39] based on the implementation by [23].

For chi-square, a threshold of 0.01 for the chi-square value was used, resulting in 1959 features for the original dataset. After applying the proposed method and the chi-square feature selection, there were 1982 features – the proposed method was applied first, resulting in different textual instances being used (i.e. the output of Algorithm 2 rather than the original dataset); consequently, when chi-square is applied after the proposed method, it results in a different number of features.

When using entropy, a threshold of 0.3 for the entropy value was employed, which led to 2634 features for both the original dataset and the dataset resulting from the proposed method.

[3]textual instances are typically referred to as documents and the dataset of all textual instances is typically referred to as the corpus

TABLE IV
FEATURE COUNT

Method	Baseline Features	Proposed Method Features
Bag-of-words	2638	2638
Doc2Vec	200	200
Chi-square	1959	1982
Entropy	2634	2634
Low Variance	1059	1042
Firefly	2500	2500
LDA	1500	1500
PCA	914	910

A threshold of 0.01 for variance was used for the low variance method, leading to 1059 features for the original dataset and 1042 features when the proposed method and low variance were applied.

For the firefly algorithm, the fitness function used is mean (μ) with light absorbance γ of 1, step size α of 0.5, maximum iteration of 10 with 2638 fireflies. We experimented with different numbers of features; here we are reporting the best performance, which was obtained when using 2500 features.

Two feature extraction methods were also employed: Latent Dirichlet Allocation (LDA) and Principal Component Analysis (PCA). LDA is typically used for topic modelling, but has also been used to extract features for text classification as mentioned in Section II. Similar to the feature selection methods, they were applied to the preprocessed original dataset and the preprocessed output of Algorithm 2.

For LDA, 10 topics and 150 terms per topic were used, resulting in 1500 features for both the original dataset and the dataset stemming from the proposed approach, as the same parameters (i.e. number of topics and number of terms per topic) were used.

For PCA, a threshold of 99% information preservation was used on both datasets, leading to 914 features for the original dataset and 910 for the output of Algorithm 2.

The number of features for each of the above methods when using the original dataset and the dataset resulting from the proposed approach are summarised in Table IV.

D. Classification and evaluation

For classification, five classifiers that have been shown to work well on textual data have been used: Decision Tree (DT), k-nearest neighbour (kNN), Naïve Bayes (NB), Random Forest (RF) and Support Vector Machine (SVM).

For all experiments, 10-fold cross-validation is employed and the Precision (P), Recall (R) and F-measure (F) are reported[4]. The results are presented and discussed in the following section.

V. Results and Discussion

Table V shows the results for the original dataset, while Table VI displays the results for the dataset stemming from the proposed method.

[4]Accuracy is not reported, as it tends to be misleading when there is a class-imbalance in the distribution of instances

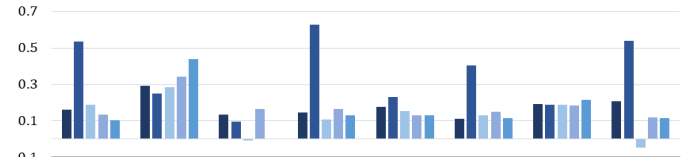
TABLE V
BASELINE RESULTS

		DT			KNN			NB			RF			SVM		
		P	R	F	P	R	F	P	R	F	P	R	F	P	R	F
Bag-of-words	Target	0.662	0.642	0.652	0.173	0.767	0.283	0.464	0.855	0.602	0.745	0.660	0.700	0.661	0.774	0.713
	Other	0.937	0.942	0.940	0.897	0.356	0.510	0.970	0.826	0.892	0.941	0.960	0.951	0.959	0.930	0.944
	F-Measure		0.796			0.396			0.747			0.825			0.829	
Doc2Vec	Target	0.525	0.522	0.524	0.536	0.698	0.607	0.735	0.157	0.259	0.768	0.396	0.523	0.000	0.000	0.000
	Other	0.916	0.917	0.917	0.944	0.894	0.918	0.870	0.990	0.926	0.902	0.979	0.939	0.851	1.000	0.919
	F-Measure		0.720			0.762			0.593			0.731			0.460	
Chi-square	Target	0.511	0.329	0.400	0.282	0.629	0.389	0.226	0.757	0.348	0.909	0.143	0.247	0.667	0.057	0.105
	Other	0.945	0.973	0.959	0.965	0.864	0.912	0.974	0.779	0.866	0.932	0.999	0.964	0.926	0.998	0.960
	F-Measure		0.679			0.651			0.607			0.606			0.533	
Entropy	Target	0.648	0.677	0.663	0.144	0.772	0.243	0.330	0.930	0.488	0.711	0.576	0.636	0.641	0.778	0.703
	Other	0.943	0.936	0.939	0.833	0.199	0.322	0.982	0.670	0.797	0.928	0.959	0.943	0.960	0.924	0.941
	F-Measure		0.801			0.282			0.642			0.790			0.822	
Low Variance	Target	0.608	0.654	0.630	0.325	0.623	0.427	0.523	0.843	0.646	0.750	0.660	0.702	0.614	0.730	0.667
	Other	0.938	0.926	0.932	0.921	0.772	0.840	0.969	0.865	0.914	0.941	0.961	0.951	0.951	0.919	0.935
	F-Measure		0.781			0.633			0.780			0.827			0.801	
Firefly	Target	0.626	0.704	0.663	0.174	0.730	0.281	0.405	0.874	0.554	0.720	0.535	0.614	0.609	0.755	0.674
	Other	0.947	0.926	0.936	0.891	0.389	0.542	0.972	0.774	0.862	0.922	0.963	0.942	0.955	0.915	0.934
	F-Measure		0.799			0.411			0.708			0.778			0.804	
LDA	Target	0.623	0.604	0.613	0.538	0.358	0.430	0.533	0.818	0.645	0.764	0.528	0.625	0.675	0.497	0.572
	Other	0.931	0.936	0.933	0.893	0.946	0.919	0.965	0.874	0.917	0.921	0.971	0.946	0.915	0.958	0.936
	F-Measure		0.773			0.675			0.781			0.785			0.754	
PCA	Target	0.546	0.560	0.553	0.169	0.748	0.276	0.736	0.560	0.636	0.890	0.459	0.606	0.626	0.767	0.689
	Other	0.922	0.918	0.920	0.889	0.353	0.505	0.926	0.965	0.945	0.912	0.990	0.950	0.957	0.919	0.938
	F-Measure		0.736			0.390			0.790			0.778			0.814	

TABLE VI
PROPOSED METHOD RESULTS

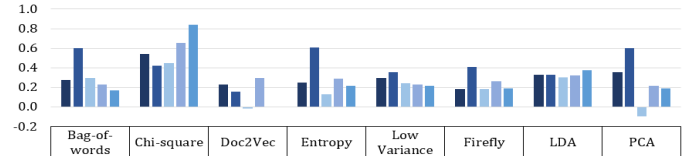
		DT			KNN			NB			RF			SVM		
		P	R	F	P	R	F	P	R	F	P	R	F	P	R	F
Bag-of-words	Target	0.876	0.981	0.925	0.891	0.880	0.885	0.810	1.000	0.895	0.959	0.899	0.928	0.890	0.873	0.882
	Other	0.997	0.976	0.986	0.979	0.981	0.980	1.000	0.959	0.979	0.982	0.993	0.988	0.978	0.981	0.980
	F-Measure		0.956			0.933			0.937			0.958			0.931	
Doc2Vec	Target	0.727	0.786	0.755	0.673	0.881	0.763	0.917	0.138	0.240	0.902	0.755	0.822	0.000	0.000	0.000
	Other	0.962	0.948	0.955	0.978	0.925	0.951	0.868	0.998	0.929	0.958	0.986	0.972	0.851	1.000	0.919
	F-Measure		0.855			0.857			0.584			0.897			0.460	
Chi-square	Target	1.000	0.893	0.943	1.000	0.679	0.809	0.675	0.964	0.794	1.000	0.821	0.902	1.000	0.893	0.943
	Other	0.996	1.000	0.998	0.989	1.000	0.994	0.999	0.984	0.991	0.994	1.000	0.997	0.996	1.000	0.998
	F-Measure		0.971			0.901			0.893			0.949			0.971	
Entropy	Target	0.958	0.867	0.910	0.810	0.892	0.849	0.451	0.994	0.621	0.959	0.892	0.925	0.979	0.867	0.919
	Other	0.977	0.993	0.985	0.981	0.963	0.972	0.999	0.788	0.881	0.981	0.993	0.987	0.977	0.997	0.987
	F-Measure		0.948			0.911			0.751			0.956			0.953	
Low Variance	Target	0.876	0.987	0.929	0.641	0.994	0.779	0.802	1.000	0.890	0.942	0.918	0.929	0.881	0.886	0.883
	Other	0.998	0.976	0.987	0.999	0.903	0.948	1.000	0.957	0.978	0.986	0.990	0.988	0.980	0.979	0.980
	F-Measure		0.958			0.864			0.934			0.959			0.931	
Firefly	Target	0.839	0.854	0.846	0.644	0.734	0.686	0.587	1.000	0.740	0.900	0.854	0.877	0.862	0.867	0.864
	Other	0.974	0.971	0.973	0.952	0.929	0.941	1.000	0.877	0.935	0.975	0.983	0.979	0.977	0.976	0.976
	F-Measure		0.910			0.814			0.837			0.928			0.920	
LDA	Target	0.938	0.949	0.943	0.897	0.658	0.759	0.949	0.949	0.949	0.943	0.949	0.946	0.944	0.956	0.950
	Other	0.991	0.989	0.990	0.943	0.987	0.964	0.991	0.991	0.991	0.991	0.990	0.991	0.992	0.990	0.991
	F-Measure		0.967			0.862			0.970			0.968			0.970	
PCA	Target	0.900	0.911	0.906	0.890	0.867	0.878	0.840	0.399	0.541	0.982	0.703	0.819	0.885	0.873	0.879
	Other	0.984	0.982	0.983	0.977	0.981	0.979	0.904	0.987	0.943	0.950	0.998	0.974	0.978	0.980	0.979
	F-Measure		0.945			0.929			0.742			0.896			0.929	

Figs 3 and 4 summarise the difference between the baseline results and the proposed method results in terms of the F-measure. Fig. 3 shows the difference in F-measure for both classes (i.e. the overall F-measure), while Fig. 4 shows the difference in F-measure for the target class.



	Bag-of-words	Chi-square	Doc2Vec	Entropy	Low Variance	Firefly	LDA	PCA
DT	0.160	0.291	0.135	0.147	0.176	0.110	0.193	0.208
KNN	0.537	0.251	0.094	0.628	0.230	0.402	0.187	0.538
NB	0.190	0.286	-0.009	0.109	0.154	0.129	0.189	-0.048
RF	0.133	0.343	0.166	0.166	0.132	0.150	0.183	0.119
SVM	0.102	0.438	0.000	0.131	0.130	0.116	0.216	0.115

Fig. 3. Impact of proposed method on both classes (F-measure)



	Bag-of-words	Chi-square	Doc2Vec	Entropy	Low Variance	Firefly	LDA	PCA
DT	0.274	0.543	0.232	0.248	0.298	0.184	0.330	0.353
KNN	0.602	0.419	0.156	0.606	0.352	0.406	0.329	0.602
NB	0.293	0.447	-0.019	0.133	0.244	0.186	0.304	-0.095
RF	0.228	0.655	0.299	0.288	0.227	0.263	0.322	0.213
SVM	0.169	0.838	0.000	0.217	0.216	0.190	0.377	0.190

Fig. 4. Impact of proposed method on the target class (F-measure)

For both the target class and the overall F-measure, the performance has improved in 37 of 40 cases. The three cases where the performance did not improve are: (a) the SVM classifier when using the doc2vec representation; with and without our method the results are the same; (b) the Naïve Bayes classifier when using the doc2vec feature representation, where there is a decrease of less than 1% in the overall F-measure and a decrease of less than 2% for the target class F-measure; (c) Naïve Bayes classifier when using PCA for feature extraction, where for both the overall F-measure and the target class F-measure, the decrease is less than 10%.

From the 37 cases where there is an improvement with the proposed method, the overall F-measure is improved by up to 10% for one case, between 10% and 20% in 24 cases, between 20% and 30% in 6 cases and by more than 30% in 6 cases. For the target class, there is an improvement between 10% and 20% in 7 cases, between 20% and 30% in 14 cases, and over 30% in 16 cases.

The best average improvement in performance across all classifiers is obtained when using the chi-square feature selection for both the overall F-measure (with an average improvement of 32%) and the target class F-measure (with an average improvement of 58%). The next best average improvement is obtained with the bag-of-words representation for the overall F-measure (average improvement of 22%) and with the LDA feature extraction for the target class F-measure (average improvement of 33%).

Although improvements can be observed across all five classifiers, some of these led to larger improvements, especially for the target class. KNN has the larger improvement in 5 out of 8 cases[5], followed by SVM with the largest improvement in 2 cases.

In terms of the best combinations of classifiers and feature representation/feature selection/feature extraction methods, for the overall F-measure, the largest improvement is obtained when using KNN with entropy (63%), KNN with PCA (54%) and KNN with bag-of-words (54%). For the target class F-measure, the largest improvement is obtained when using SVM with chi-square (84%), followed by the same three combinations as for the overall F-measure (with 61%, 60% and 60% improvement, respectively).

In summary, the results show that the proposed method leads to an increase in performance overall and, in particular, for the target class. This increase occurs with and without the use of feature selection or feature extraction methods, although in particular cases (e.g. chi-square and LDA), these boost the performance significantly for the target class.

VI. Conclusions and Future Work

In this paper, we proposed a method for the disambiguation of features, with the aim to improve the classification performance for target classes from social media text.

The method was presented for a binary class problem and involved the allocation of terms to the target and other class, followed by the removal of these terms from instances of the opposite class (i.e. terms from the target class are removed from instances of the other class).

The method was validated using a dataset for suicide detection, employing five classifiers, two methods for feature representation, four feature selection methods and two feature extraction methods. The results showed an improvement in 37 of 40 cases, and an improvement of more than 20% in the F-measure of the target class for 75% of the cases (i.e. 30 of 40).

Although the method was presented for a binary classification problem, it can be adjusted for a multi-class classification problem with multiple target and other classes. This is part of our future work, as well as investigating the use of the proposed method in combination with methods for addressing class imbalance, given that often target classes are also minority classes.

Acknowledgment

This research was supported by the Petroleum Development Technology Fund (PTDF) and the Department of Health Policy Research Programme (Understanding the Role of Social Media in the Aftermath of Youth Suicides, Project Number 023/0165). The views expressed in this publication are those of the authors and not necessarily those of PTDF or the Department of Health.

References

[1] S. Agrawal and A. Awekar, "Deep learning for detecting cyberbullying across multiple social media platforms," in *European Conference on Information Retrieval*. Springer, 2018, pp. 141–153.

[5]the 8 cases are the 2 feature representations, the 4 feature selection and the 2 feature extraction methods

[2] M. A. Al-garadi, K. D. Varathan, and S. D. Ravana, "Cybercrime detection in online communications: The experimental case of cyberbullying detection in the Twitter network," *Computers in Human Behavior*, vol. 63, pp. 433–443, 2016.

[3] N. Alkiviadou, "Hate speech on social media networks: towards a regulatory framework?" *Information & Communications Technology Law*, vol. 28, no. 1, pp. 19–35, 2019.

[4] V. Banerjee, J. Telavane, P. Gaikwad, and P. Vartak, "Detection of cyberbullying using deep neural network," in *2019 5th International Conference on Advanced Computing & Communication Systems (ICACCS)*. IEEE, 2019, pp. 604–607.

[5] A. Bhati and D. McDonnell, "Success in an online giving day: The role of social media in fundraising," *Nonprofit and Voluntary Sector Quarterly*, vol. 49, no. 1, pp. 74–92, 2020.

[6] P. Burnap, G. Colombo, R. Amery, A. Hodorog, and J. Scourfield, "Multi-class machine classification of suicide-related communication on twitter," *Online social networks and media*, vol. 2, pp. 32–44, 2017.

[7] P. Burnap, W. Colombo, and J. Scourfield, "Machine classification and analysis of suicide-related communication on Twitter," in *Proceedings of the 26th ACM conference on hypertext & social media*. ACM, 2015, pp. 75–84.

[8] F. Chiroma, M. Cocea, and H. Liu, "Detection of suicidal Twitter posts," in *UK Workshop on Computational Intelligence*. Springer, 2019, pp. 307–318.

[9] ——, "Evaluation of rule-based learning and feature selection approaches for classification," in *2018 Imperial College Computing Student Workshop (ICCSW 2018)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2019.

[10] G. Coppersmith, R. Leary, P. Crutchley, and A. Fine, "Natural language processing of social media as screening for suicide risk," *Biomedical informatics insights*, vol. 10, pp. 1–F11, 2018.

[11] R. Delgado, A. K. Wing, and J. Stefancic, "Words that wound: A tort action for racial insults, epithets, and name-calling," in *Law Unbound!* Routledge, 2015, pp. 223–228.

[12] X. Deng, Y. Li, J. Weng, and J. Zhang, "Feature selection for text classification: A review," *Multimedia Tools and Applications*, vol. 78, no. 3, pp. 3797–3816, 2019.

[13] X. Huang, L. Zhang, D. Chiu, T. Liu, X. Li, and T. Zhu, "Detecting suicidal ideation in Chinese microblogs with psychological lexicons," in *2014 IEEE 11th Intl Conf on Ubiquitous Intelligence and Computing and 2014 IEEE 11th Intl Conf on Autonomic and Trusted Computing and 2014 IEEE 14th Intl Conf on Scalable Computing and Communications and Its Associated Workshops*. IEEE, 2014, pp. 844–849.

[14] S. Ji, C. P. Yu, S.-f. Fung, S. Pan, and G. Long, "Supervised learning for suicidal ideation detection in online user content," *Complexity*, vol. 2018, 2018, article ID 6157249. [Online]. Available: https://doi.org/10.1155/2018/6157249

[15] C. Kennedy, "Ambiguity and vagueness: An overview," in *Semantics-Lexical Structures and Adjectives*. Walter de Gruyter GmbH & Co KG, 2019, pp. 236–271.

[16] I. Kwok and Y. Wang, "Locate the hate: Detecting tweets against blacks," in *Proceedings of the twenty-seventh AAAI conference on Artificial Intelligence*, 2013, pp. 1621–1622.

[17] Q. Le and T. Mikolov, "Distributed representations of sentences and documents," in *Proceedings of the 31st International conference on machine learning*, 2014, pp. 1188–1196.

[18] J. Li, K. Cheng, S. Wang, F. Morstatter, R. P. Trevino, J. Tang, and H. Liu, "Feature selection: A data perspective," *ACM Computing Surveys (CSUR)*, vol. 50, no. 6, pp. 1–45, 2017.

[19] V. Lingiardi, N. Carone, G. Semeraro, C. Musto, M. D'Amico, and S. Brena, "Mapping Twitter hate speech towards social and sexual minorities: a lexicon-based approach to semantic content analysis," *Behaviour & Information Technology*, pp. 1–11, 2019.

[20] H. Liu, P. Burnap, W. Alorainy, and M. L. Williams, "A fuzzy approach to text classification with two-stage training for ambiguous instances," *IEEE Transactions on Computational Social Systems*, vol. 6, no. 2, pp. 227–240, 2019.

[21] Y. Liu, P. Zavarsky, and Y. Malik, "Non-linguistic features for cyberbullying detection on a social media platform using machine learning," in *International Symposium on Cyberspace Safety and Security*. Springer, 2019, pp. 391–406.

[22] J. Lopez-Castroman, B. Moulahi, J. Azé, S. Bringay, J. Deninotti, S. Guillaume, and E. Baca-Garcia, "Mining social networks to improve suicide prevention: A scoping review," *Journal of neuroscience research*, pp. 1–10, 2019. [Online]. Available: https://doi.org/10.1002/jnr.24404

[23] A. Manthiri S, "Firefly feature selection and optimization," 2017. [Online]. Available: https://www.mathworks.com/matlabcentral/fileexchange/62235-firefly-feature-selection-and-optimization

[24] M. Marcus, G. Kim, M. A. Marcinkiewicz, R. MacIntyre, A. Bies, M. Ferguson, K. Katz, and B. Schasberger, "The Penn Treebank: annotating predicate argument structure," in *Proceedings of the workshop on Human Language Technology*. Association for Computational Linguistics, 1994, pp. 114–119.

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

[26] S. A. Özel, E. Saraç, S. Akdemir, and H. Aksu, "Detection of cyberbullying on social media messages in Turkish," in *2017 International Conference on Computer Science and Engineering (UBMK)*. IEEE, 2017, pp. 366–370.

[27] L. Palen and A. L. Hughes, "Social media in disaster communication," in *Handbook of disaster research*. Springer, 2018, pp. 497–518.

[28] G. K. Pitsilis, H. Ramampiaro, and H. Langseth, "Effective hate-speech detection in Twitter data using recurrent neural networks," *Applied Intelligence*, vol. 48, no. 12, pp. 4730–4742, 2018.

[29] P. Pond and J. Lewis, "Riots and Twitter: Connective politics, social media and framing discourses in the digital public sphere," *Information, Communication & Society*, vol. 22, no. 2, pp. 213–231, 2019.

[30] R. Sawhney, P. Manchanda, P. Mathur, R. Shah, and R. Singh, "Exploring and learning suicidal ideation connotations on social media with deep learning," in *Proceedings of the 9th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, 2018, pp. 167–175.

[31] R. Sawhney, P. Manchanda, R. Singh, and S. Aggarwal, "A computational approach to feature extraction for identification of suicidal ideation in tweets," in *Proceedings of ACL 2018, Student Research Workshop*, 2018, pp. 91–98.

[32] R. Sawhney, R. R. Shah, V. Bhatia, C.-T. Lin, S. Aggarwal, and M. Prasad, "Exploring the impact of evolutionary computing based feature selection in suicidal ideation detection," in *2019 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*. IEEE, 2019, pp. 1–6.

[33] R. Sedgwick, S. Epstein, R. Dutta, and D. Ougrin, "Social media, internet use and suicide attempts in adolescents," *Current opinion in psychiatry*, vol. 32, no. 6, p. 534, 2019.

[34] H.-C. Shing, S. Nair, A. Zirikly, M. Friedenberg, H. Daumé III, and P. Resnik, "Expert, crowdsourced, and machine assessment of suicide risk via online postings," in *Proceedings of the Fifth Workshop on Computational Linguistics and Clinical Psychology: From Keyboard to Clinic*, 2018, pp. 25–36.

[35] D. Uludasdemir and S. Kucuk, "Cyber bullying experiences of adolescents and parental awareness: Turkish example," *Journal of pediatric nursing*, vol. 44, pp. e84–e90, 2019.

[36] Z. Waseem and D. Hovy, "Hateful symbols or hateful people? Predictive features for hate speech detection on Twitter," in *Proceedings of the NAACL student research workshop*, 2016, pp. 88–93.

[37] H. Watanabe, M. Bouazizi, and T. Ohtsuki, "Hate speech on Twitter: A pragmatic approach to collect hateful and offensive expressions and perform hate speech detection," *IEEE Access*, vol. 6, pp. 13 825–13 835, 2018.

[38] G. Xiang, B. Fan, L. Wang, J. Hong, and C. Rose, "Detecting offensive tweets via topical feature discovery over a large scale Twitter corpus," in *Proceedings of the 21st ACM international conference on Information and knowledge management*. ACM, 2012, pp. 1980–1984.

[39] X.-S. Yang and X. He, "Firefly algorithm: recent advances and applications," *International Journal of Swarm Intelligence*, vol. 1, no. 1, pp. 36–50, 2013.

[40] M. Yao, C. Chelmis, D. Zois *et al.*, "Cyberbullying ends here: Towards robust detection of cyberbullying in social media," in *The World Wide Web Conference*. ACM, 2019, pp. 3427–3433.

[41] Z. Zhang, D. Robinson, and J. Tepper, "Detecting hate speech on Twitter using a convolution-GRU based deep neural network," in *European Semantic Web Conference*. Springer, 2018, pp. 745–760.