A Comprehensive Analysis of Preprocessing for Word Representation Learning in Affective Tasks

Nastaran Babanejad, Ameeta Agrawal, Aijun An, Manos Papagelis
Department of Electrical Engineering and Computer Science,
York University, Toronto, Canada
{nasba, ameeta, aan, papaggel}@eecs.yorku.ca

Abstract

Affective tasks such as sentiment analysis, emotion classification and sarcasm detection have been popular in recent years due to abundance of user-generated data, accurate computational linguistic models, and broad range of relevant applications in various domains. At the same time, many studies have highlighted the importance of text preprocessing, as an integral step to any natural language processing prediction model and downstream task. While preprocessing in affective systems is well-studied, preprocessing in word vector based models applied to affective systems, is not. To address this limitation, we conduct a comprehensive analysis of the role of preprocessing techniques in affective analysis based on word vector models. Our analysis is the first of its kind and provides useful insights of the importance of each preprocessing technique when applied at the training phase, commonly ignored in pretrained word vector models, and/or at the downstream task phase.

1 Introduction

Affective tasks such as sentiment analysis, emotion classification and sarcasm detection have enjoyed great popularity in recent years. This success can be largely attributed to the fundamental and straightforward nature of the methods employed, the availability of vast amounts of user-generated natural language data, and the wide range of useful applications, spanning from hate speech detection to monitoring the sentiment of financial markets and news recommendation (Djuric et al., 2015; Babanejad et al., 2019). Most early models of affect analysis employed pretrained word embeddings that have been obtained under the assumption of the distributional hypothesis (Mikolov et al., 2013; Devlin et al., 2018). The distributional hypothesis suggests that two words occurring frequently in similar linguistic contexts tend to be more semantically similar, and therefore should be represented closer to one another in the embedding space. However, while such embeddings are useful for several natural language processing (NLP) downstream tasks, they are known to be less suitable for affective tasks in particular (Tang et al., 2014; Agrawal et al., 2018). Although some authors claim that there is a need for post-processing word embeddings for affective tasks, others find that off-the-shelf vectors are very powerful for affective lexicon learning (Lison and Kutuzov, 2017). For example, `word2vec` (Mikolov et al., 2013) estimates the pair of words 'happy' and 'sad' to be more similar than the pair of words 'happy' and 'joy', which is counterintuitive, and might affect the accuracy performance of the models that depend on it.

To address the limitations of traditional word embeddings, several techniques have been proposed, including task-specific fine-tuning (Devlin et al., 2018), retrofitting (Faruqui et al., 2014), representing emotion with vectors using a multi-task training framework (Xu et al., 2018) and generating affective word embeddings (Felbo et al., 2017), to name a few. Other attempts to overcome the limitation of word vectors include optimization of hyperparameters (Levy et al., 2015), as well as fine-tuned preprocessing strategies tailored to different NLP tasks. While these strategies have demonstrated evidence of improving the accuracy performance in tasks such as word similarity, word analogy, and others (Lison and Kutuzov, 2017), their effect in affective tasks has not received considerable attention and remains less explored. Our work is motivated by the observation that preprocessing factors such as stemming, stopwords removal and many others make up an integral part of nearly every improved text classification model, and affective systems in particular (Danisman and Alpkocak, 2008; Patil and Patil, 2013). However, little work has been

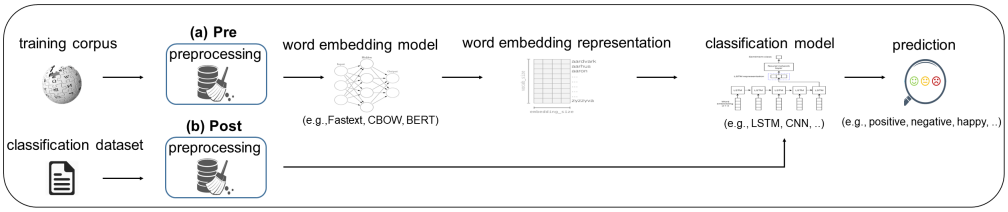

Figure 1: Framework of applying preprocessing in different stages in affective systems; (a) Pre, (b) Post.

done towards understanding the role of preprocessing techniques applied to word embeddings in different stages of affective systems. To address this limitation, the overarching goal of this research, is to perform an extensive and systematic assessment of the effect of a range of linguistic preprocessing factors pertaining to three affective tasks, including *sentiment analysis*, *emotion classification* and *sarcasm detection*. Towards that end, we systematically analyze the effectiveness of applying preprocessing to large training corpora *before* learning word embeddings, an approach that has largely been overlooked by the community. We investigate the following research questions: *(i)* what is the effect of integrating preprocessing techniques earlier into word embedding models, instead of later on in a downstream classification models? *(ii)* which preprocessing techniques yield the most benefit in affective tasks? *(iii)* does preprocessing of word embeddings provide any improvement over state-of-the-art pretrained word embeddings? and if yes, how much?

Figure 1 illustrates the difference between a) preprocessing word embeddings pipeline (Pre) vs. b) preprocessing classification dataset pipeline (Post), where preprocessing techniques in (a) are applied to the training corpus of the model and in (b) only to the classification dataset. In brief, the main contributions of our work are as follows:

- We conduct a comprehensive analysis of the role of preprocessing techniques in affective tasks (including sentiment analysis, emotion classification and sarcasm detection), employing different models, over nine datasets;
- We perform a comparative analysis of the accuracy performance of word vector models when preprocessing is applied at the training phase (training data) and/or at the downstream task phase (classification dataset). Interestingly, we obtain best results when preprocessing is applied only to the training corpus or when it is applied to both the training corpus and the classification dataset of interest.
- We evaluate the performance of our best preprocessed word vector model against state-of-the-art pretrained word embedding models;
- We make *source code* and *data* publicly available to encourage reproducibility of results[1].

The rest of the paper is organized as follows: Section 2 presents an overview of the related work. Section 3 elaborates on the preprocessing techniques employed in the evaluation of models. Section 4 describes the experimental evaluation framework. In Section 5 a comprehensive analysis of the results is provided. Section 6 concludes the paper with key insights of the research.

2 Related Work

In this section, we present an overview of related work on *preprocessing classification datasets* and *preprocessing word embeddings*, and how our work aims to bridge the gap between those efforts.

2.1 Preprocessing Classification Datasets

Preprocessing is a vital step in text mining and therefore, evaluation of preprocessing techniques has long been a part of many affective systems. Saif et al. (2014) indicated that, despite its popular use in Twitter sentiment analysis, the use of precompiled stoplist has a negative impact on the classification performance. Angiani et al. (2016) analyzed various preprocessing methods such as stopwords removal, stemming, negation, emoticons, and so on, and found stemming to be most effective for the task of sentiment analysis. Similarly, Symeonidis et al. (2018) found that lemmatization increases accuracy. Jianqiang and Xiaolin (2017) observed that removing stopwords, numbers, and URLs can reduce noise but does not affect performance, whereas replacing negation and expanding acronyms can improve the classification accuracy.

[1] https://github.com/NastaranBa/preprocessing-for-word-representation

Preprocessing techniques such as punctuation and negation (Rose et al., 2018) or pos-tagging and negation (Seal et al., 2020) make up a common component of many emotion classification models (Kim et al., 2018; Patil and Patil, 2013). One of the earliest works (Danisman and Alpkocak, 2008) preserved emotion words and negative verbs during stopwords removal, replaced punctuation with descriptive new words, replaced negative short forms with long forms, and concatenated negative words with emotion words to create new words (e.g., not happy → NOThappy). Although stemming may remove the emotional meaning from some words, it has been shown to improve classification accuracy (Danisman and Alpkocak, 2008; Agrawal and An, 2012). Negations have also been found beneficial, whereas considering intensifiers and diminishers did not lead to any improvements (Strohm, 2017).

Pecar et al. (2018) also highlight the importance of preprocessing when using user-generated content, with emoticons processing being the most effective. Along the same lines, while Gratian and Haid (2018) found pos-tags to be useful, Boiy et al. (2007) ignored pos-tagging because of its effect of reducing the classification accuracy

The aforementioned works describe preprocessing techniques as applied directly to evaluation datasets in affective systems. In contrast, we examine the effectiveness of directly incorporating these known effective preprocessing techniques further "upstream" into the training corpus of word embeddings, which are widely used across a number of downstream tasks.

2.2 Preprocessing Word Embeddings

Through a series of extensive experiments, particularly those related to context window size and dimensionality, (Levy et al., 2015) indicate that seemingly minor variations can have a large impact on the success of word representation methods in similarity and analogy tasks, stressing the need for more analysis of often ignored preprocessing settings. Lison and Kutuzov (2017) also present a systematic analysis of context windows based on a set of four hyperparameters, including window position and stopwords removal, where the right window was found to be better than left for English similarity task, and stopwords removal substantially benefited analogy task but not similarity.

A general space of hyperparameters and preprocessing factors such as context window size (Hershcovich et al., 2019; Melamud et al., 2016), dimensionality (Melamud et al., 2016), syntactic dependencies (Levy and Goldberg, 2014; Vulić et al., 2020) and their effect on NLP tasks including word similarity (Hershcovich et al., 2019), tagging, parsing, relatedness, and entailment (Hashimoto et al., 2017) and biomedical (Chiu et al., 2016) has been studied extensively in the literature. The main conclusion of these studies, however, is that these factors are heavily task-specific. Therefore, in this work we explore preprocessing factors of generating word embeddings specifically tailored to affective tasks, which have received little attention.

A recent study investigated the role of tokenizing, lemmatizing, lowercasing and multiword grouping (Camacho-Collados and Pilehvar, 2018) as applied to sentiment analysis and found simple tokenization to be generally adequate. In the task of emotion classification, Mulki et al. (2018) examined the role of four preprocessing techniques as applied to a vector space model based on tf-idf trained on a small corpus of tweets, and found stemming, lemmatization and emoji tagging to be the most effective factors.

Distinct from prior works, we examine a much larger suite of preprocessing factors grounded in insights derived from numerous affective systems, trained over two different corpora, using three different word embedding models. We evaluate the effect of the preprocessed word embeddings in three distinct affective tasks including sentiment analysis, emotion classification and sarcasm detection.

3 Preprocessing in Affective Systems

This section describes the preprocessing factors applied to the training corpus that is then used to generate word representations and the order of the preprocessing factors which we need to follow when applying on the corpus.

3.1 Preprocessing Factors

Basic: A group of common text preprocessing applied at the very beginning, such as removing html tags, removing numbers, and lowercasing. This step removes all common punctuation from text, such as "@%*=()/ +" using the NLTK regexptokenizer[2].

Spellcheck (`spell`): A case can be made for either correcting misspellings and typos or leaving

[2]https://www.nltk.org/_modules/nltk/tokenize/regexp.html

them as is assuming they represent natural language text and its associated complexities. In this step, we identify words that may have been misspelled and correct them[3]. As unambiguous spell corrections are not very common and in most cases we have multiple options for correction, we built our own custom dictionary to suggest a replacement by parsing the ukWac corpora[4] to retrieve a word-frequency list. A misspelled word that has multiple replacements is replaced with the suggested word that has the maximum frequency in the corpora.

Negation (`neg`): Negation is a mechanism that transforms a positive argument into its inverse rejection (Benamara et al., 2012). Specifically in the task of affective analysis, negation plays a critical role as negation words can affect the word or sentence polarity causing the polarity to invert in many cases. Our negation procedure is as follows:
(i) Compilation of an antonym dictionary: The first stage involves compiling an antonym dictionary using the WordNet corpus (Miller, 1995). For every synset, there are three possibilities: finding no antonym, one antonym or multiple antonyms. The first two cases are trivial (unambiguous replacements). In the case of the third option (ambiguous replacement), which represents the most common case, amongst the many choices, we consider the antonym with the maximum frequency in the ukWac corpus, as described in the previous section and finally the antonym of a word is picked at random from one of its senses in our antonym dictionary.
(ii) Negation handler: Next, we identify the negation words in tokenized text[5]. If a negation word is found, the token following it (i.e., negated word) is extracted and its antonym looked up in the antonym dictionary. If an antonym is found, the negation word and the negated word are replaced with it.

For example, let the sentence "I am not happy today" in its tokenized form ['I', 'am', 'not', 'happy', 'today']. First, we identify any negation words (i.e., 'not') and their corresponding negated words (i.e., 'happy'). Then, we look up the antonym of 'happy' in the antonym dictionary (i.e., 'sad') and replace the phrase 'not happy' with the word 'sad', resulting in a new sentence "I am sad today".

Parts-of-Speech (`pos`): Four parts-of-speech classes, namely nouns, verbs, adjectives and adverbs have been shown to be more informative with regards to affect than the other classes. Thus, using the NLTK pos-tagger, for each sentence in the corpus we retain only the words belonging to one of these four classes, i.e., NN*, JJ*, VB*, and RB*.

Stopwords (`stop`): Stopwords are generally the most common words in a language typically filtered out before classification tasks. Therefore, we remove all the stopwords using the NLTK library.

Stemming (`stem`): Stemming, which reduces a word to its root form, is an essential preprocessing technique in NLP tasks. We use NLTK Snowball stemmer for stemming our training corpus.

3.2 Order of Preprocessing Factors

While some preprocessing techniques can be applied independently of each other (e.g., removing stopwords and removing punctuation), others need a more careful consideration of the sequence in which they are applied in order to obtain a more stable result. For instance, pos-tagging should be applied before stemming in order for the tagger to work well, or negation should be performed prior to removing stopwords. To this end, we consider the following ordering when combining all the aforementioned preprocessing factors: spellchecking, negation handling, pos classes, removing stopwords, and stemming.

4 Experimental Evaluation Framework

4.1 Training Corpora

Table 1 summarizes the details of our two training corpora with regards to their vocabulary and corpus sizes after applying various preprocessing settings. For some preprocessing such as POS (`pos`) and stopwords removal (`stop`), without any significant loss in vocabulary as indicated by the % ratio of preprocessed to basic, the corpus size reduces dramatically, in some cases more than 50%, a non-trivial implication with regards to training time.
News: This corpus consists of 142,546 articles from 15 American publications, spanning from 2013 to early 2018[6].
Wikipedia: Comparatively a much larger corpus than the News, this corpus consists of 23,046,187 articles from Wikipedia [7].

[3]https://pypi.org/project/pyspellchecker/
[4]https://www.sketchengine.eu/ukwac-british-english-corpus/
[5]https://pypi.org/project/negspacy/
[6]https://www.kaggle.com/snapcrack/all-the-news
[7]https://www.kaggle.com/jkkphys/english-wikipedia-articles-20170820-sqlite

Corpus	Processing	Vocab		Corpus	
		size	%	size	%
News	Basic	155K	100	123.2M	100
	spell	149K	96	123.2M	100
	stem	137K	88	123.2M	100
	punc	147K	95	111.0M	90
	neg	152K	98	90.7M	73
	stop	150K	97	75.6M	61
	pos	154K	99	70.7M	57
	All - punc	151K	97	93.7M	76
	All - pos	140K	90	90.5M	73
	All - stop	150K	97	75.3M	61
	All	110K	71	55.2M	49
	All - stem	110K	71	58.1M	47
	All - spell	110K	71	56.4M	46
	All - neg	110K	71	54.3M	44
Wikipedia	Basic	5.1M	100	8.1B	100
	All - punc	4.9M	96	7.2B	89
	All - pos	4.8M	94	7.0B	86
	All - stop	4.9M	96	6.8B	84
	All - stem	4.3M	84	6.4B	79
	All - spell	4.6M	90	6.1B	75
	All	4.6M	90	5.6B	69
	All - neg	4.6M	90	5.0B	62

Table 1: Details of training corpora

Dataset	Genre	Task	Total
IMDB	reviews	sentiment	50,000
SemEval	tweets	sentiment	14,157
Airline	tweets	sentiment	11,541
ISEAR	narratives	emotions	5,477
Alm	fairy tales	emotions	1,206
SSEC	tweets	emotions	1,017
Onion	headlines	sarcasm	28,619
IAC	response	sarcasm	3,260
Reddit	comments	sarcasm	1,010,826

Table 2: Details of evaluation datasets

4.2 Word Embedding Models

We obtain our preprocessed word representations through three models: (i) **CBOW (Continuous Bag-of-Words)**, (ii) **Skip-gram**: While CBOW takes the context of each word as the input and tries to predict the word corresponding to the context, skip-gram reverses the use of target and context words, where the target word is fed at the input and the output layer of the neural network is replicated multiple times to accommodate the chosen number of context words (Mikolov et al., 2013). We train both the models on both the training corpora using min count of 5 for News and 100 for Wikipedia with window sizes of 5 and 10, respectively, setting dimensionality to 300.

(iii) **BERT (Bidirectional Encoder Representations from Transformers)**: BERT is an unsupervised method of pretraining contextualized language representations (Devlin et al., 2018). We train the model using BERT large uncased architecture (24-layer, 1024-hidden, 16-heads, 340M parameters) with same setting for parameters as the original paper.

We train each of the three models (CBOW, Skip-gram and BERT) 8 times using 16 TPUs (64 TPU chips), Tensorflow 1.15, 1TB memory on Google Cloud and two 32 GPUs cluster of V100/RTX 2080 Ti, 1TB memory using Microsoft CNTK parallelization algorithm[8] on Amazon server. For a large model such as BERT, it takes upto 4-5 days for each run of the training.

4.3 Evaluation Datasets

We conduct our evaluation on three tasks, namely sentiment analysis, emotion classification and sarcasm detection. Table 2 presents the details of our evaluation datasets, and some illustrative examples of text are shown in Table 3.

Sentiment Analysis: This popular task involves classifying text as positive or negative, and we use the following three datasets for evaluation: (i) **IMDB**: This dataset[9] includes 50,000 movie reviews for sentiment analysis, consisting of 25,000 negative and 25,000 positive reviews Maas et al. (2011). (ii) **Semeval 2016**: This sentiment analysis in Twitter dataset[10] consists of 14,157 tweets where 10,076 of them are positive and 4,081 negative Nakov et al. (2016). (iii) **Airlines**: This sentiment analysis dataset[11] consists of 11,541 tweets about six U.S. airlines from February 2015, with 9,178 tweets labeled as positive and 2,363 negative.

Emotion Classification: A multiclass classification task, this involves classifying text into a number of emotion categories such as happy, sad, and so on. The following datasets are used in our evaluation: (i) **SSEC**: The Stance Sentiment Emotion Corpus Schuff et al. (2017) is the re-annotation of the SemEval 2016 Twitter stance and sentiment corpus Mohammad et al. (2017) with emotion labels including anger, joy, sadness, fear, surprise. [12]. (ii) **ISEAR**: This dataset contains narratives of personal experiences evoking emotions Wallbott and Scherer (1986). We use a subset of the data consisting of five categories: sadness, anger, disgust, fear, joy. (iii) **Alm**: This dataset contains sentences

[8] https://docs.microsoft.com/en-us/cognitive-toolkit/multiple-gpus-and-machines

[9] http://ai.stanford.edu/ amaas/data/sentiment/

[10] http://alt.qcri.org/semeval2016/task4/index.php

[11] https://www.kaggle.com/crowdflower/twitter-airline-sentiment

[12] SSEC: http://www.romanklinger.de/ssec/

Text	Label	Dataset
· *I must admit that this is one of the worst movies I've ever seen. I thought Dennis Hopper had a little more taste than to appear in this kind of yeeeecchh... [truncated]*	negative	IMDB
· *everything was fine until you lost my bag.*	negative	Airline
· *At work, when an elderly man complained unjustifiably about me and distrusted me.*	anger	ISEAR
· *The ladies danced and clapped their hands for joy.*	happy	Alm
· *if this heat is killing me i don't wanna know what the poor polar bears are going through*	sadness	SSEC
· *ford develops new suv that runs purely on gasoline*	sarcastic	Onion
· *Been saying that ever since the first time I heard about creationsism*	not-sarcastic	IAC
· *Remember, it's never a girl's fault, it's always the man's fault.*	sarcastic	Reddit

Table 3: Examples of text instances in the evaluation datasets

from fairy tales marked with one of five emotion categories: angry-disgusted, fearful, happy, sad and surprised Cecilia and Ovesdotter (2008).

Sarcasm Detection: Detecting sarcasm from text, a challenging task due to the sophisticated nature of sarcasm, involves labeling text as sarcastic or not. We use the following three datasets: (i) **Onion**: This news headlines dataset [13] collected sarcastic versions of current events from The Onion and non-sarcastic news headlines from HuffPost Misra and Arora (2019), resulting in a total 28,619 records. (ii) **IAC**: A subset of the Internet Argument Corpus Oraby et al. (2016), this dataset contains response utterances annotated for sarcasm. We extract 3260 instances from the general sarcasm type.[14]. (iii) **Reddit**: Self-Annotated Reddit Corpus (SARC)[15] is a collection of Reddit posts where sarcasm is labeled by the author in contrast to other datasets where the data is typically labeled by independent annotators Khodak et al. (2017).

4.4 Classification Setup

For classification, we employ the LSTM model as it works well with sequential data such as text. For binary classification, such as sentiment analysis and sarcasm detection, the loss function used is the binary cross-entropy along with sigmoid activation:

$$\xi = -\frac{1}{N}\sum_{i=1}^{N} y_i log(p(y_i)) + (1-y_i) log(1-p(y_i))$$

where y is the binary representation of true label, $p(y)$ is the predicted probability, and i denotes the i^{th} training sample.

For multiclass emotion classification, the loss function used is categorical cross-entropy loss over a batch of N instances and k classes, along with softmax activation:

$$\xi = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{k} y_{ij} log\left(p(y_{ij})\right)$$

where $p(y)$ is the predicted probability distribution, $p(y_{ij}) \in [0, 1]$.

The optimizer is Adam Kingma and Ba (2014), all loss functions are sample-wise, and we take the mean of all samples (epoch $= 5$, 10, batch size $= 64$, 128). All sentiment and sarcasm datasets are split into training/testing using 80%/20%, with 10% validation from training. For the smaller and imbalanced emotion datasets, we use stratified 5-fold cross-validation. We use a dropout layer to prevent overfitting by ignoring randomly selected neurons during training. We use early stopping when validation loss stops improving with patience = 3, min-delta = 0.0001. The results are reported in terms of weighted F-score (as some emotion datasets are highly imbalanced), where F-score $= 2\frac{p.r}{p+r}$, with p denoting precision, and r is recall.

5 Discussion and Analysis

We analyze the impact of preprocessing techniques in word representation learning on affect analysis.

5.1 Effect of Preprocessing Factors

A primary goal of this work is to identify the most effective preprocessing factors for training word embeddings for affective tasks. Table 4 details the results of our experiments comparing the performance of individual preprocessing factors as well as those of ablation studies (i.e., including all the factors but one).

Observing the performance of the individual factors on the News corpus, we note that even a single simple preprocessing technique can bring improvements, thereby validating our intuition of incorporating preprocessing into training corpora of word representations. Second, negation (`neg`) processing appears to be consistently the most

[13] https://github.com/rishabhmisra/News-Headlines-Dataset-For-Sarcasm-Detection

[14] https://nlds.soe.ucsc.edu/sarcasm2

[15] SARC v0.0: https://nlp.cs.princeton.edu/SARC/0.0/

Models	Processing	IMDB	Semeval	Airline	IAC	Onion	Reddit	Alm	ISEAR	SSEC
CBOW	Basic	83.99	55.69	60.73	65.74	68.23	59.42	36.81	55.43	51.76
	stop	84.43	55.72	61.37	66.03	68.17	59.27	36.81	56.01	52.33
	spell	86.20	55.93	61.96	66.00	69.57	60.00	36.88	56.41	52.14
	stem	86.92	55.72	61.86	65.89	68.49	59.72	36.94	55.84	51.89
	punc	86.99	56.41	62.08	65.93	69.85	60.28	36.94	56.89	52.03
	pos	85.66	56.83	62.75	66.32	70.25	60.63	37.02	57.04	53.19
	neg	88.98	57.29	63.81	66.87	71.12	60.91	37.22	57.39	54.15
	All	89.96	57.82	64.58	67.23	70.90	60.84	37.43	57.72	53.71
	All - neg	84.67	55.00	61.58	66.02	69.73	59.94	36.91	55.89	51.94
	All - pos	85.69	56.31	64.29	66.97	70.48	60.15	37.19	56.27	52.16
	All - punc	86.41	56.88	63.01	66.75	70.01	60.00	37.01	57.19	52.43
	All - spell	88.23	56.41	63.87	67.23	70.83	60.27	37.22	57.41	53.41
	All - stop	**90.01**	**60.82**	66.84	67.20	**72.49**	**62.11**	**38.96**	**59.28**	55.00
	All - stem	88.12	**60.82**	**67.12**	**69.25**	72.13	61.73	38.00	59.00	**55.42**
Skip-gram	Basic	83.07	54.23	61.47	65.51	68.01	59.75	35.87	55.64	51.49
	stop	83.23	55.47	62.00	65.62	68.00	59.84	35.94	55.76	51.62
	spell	85.90	55.48	62.00	65.61	69.76	60.28	36.10	55.93	52.30
	stem	86.00	55.33	61.89	65.60	68.72	59.50	36.00	55.69	51.40
	punc	86.68	55.79	62.38	65.89	70.00	60.44	36.41	56.81	52.71
	pos	85.91	56.28	63.25	66.24	69.81	60.85	36.44	56.23	52.94
	neg	87.28	56.89	63.72	66.87	70.59	61.27	36.87	57.34	53.10
	All	88.36	57.04	64.91	66.94	70.73	61.12	37.10	57.92	53.58
	All - neg	83.26	54.00	61.95	66.00	69.88	60.00	36.94	55.97	51.89
	All - pos	86.21	55.22	65.12	66.06	69.88	61.00	37.00	56.42	52.10
	All - punc	85.57	55.99	64.29	66.29	70.00	60.98	37.01	57.02	52.53
	All - spell	86.00	56.98	65.00	66.25	70.25	0.61	37.04	57.69	52.86
	All - stop	**88.74**	**60.93**	67.00	68.57	**72.20**	62.02	**38.92**	59.18	55.18
	All - stem	88.42	60.67	**67.39**	**69.08**	72.00	**62.36**	37.44	**59.48**	**55.23**

Table 4: F-score results of evaluating the effect of preprocessing factors using CBOW and Skip-gram on News corpus. The overall best results are in **bold**. The best result using only any one preprocessing setting is underlined.

effective factor across all the 9 datasets, indicating its importance in affective classification, followed by parts-of-speech (pos) processing where we retained words belonging only to one of four classes. On the other hand, removing stopwords (stop), spellchecking (spell) and stemming (stem) yield little improvement and mixed results. Interestingly, applying all the preprocessing factors is barely better or in some cases even worse (Onion, Reddit and SSEC) than applying just negation. Finally, the best performance comes from combining all the preprocessing factors except stemming (All-stem). Moreover, Table 5 details the performance of ablation studies on Wikipedia corpus for all three models where we note that the best performance for the CBOW model comes from combining all the preprocessing factors except stemming (All-stem), whereas for the Skip-gram and BERT models, the best results are obtained by applying all the preprocessing factors except stopwords removal (All-stop). Considering that the Wikipedia corpus is almost 160 times bigger than the News corpus, it is unsurprising that the word embeddings obtained from the former yield considerably better results, consistent across all nine datasets.

5.2 Evaluating Preprocessing Training Corpora for Word Vectors vs. Preprocessing Classification Data

We investigate the difference between applying preprocessing to the training corpora for generating word embeddings (Pre) and applying preprocessing to the classification datasets (Post). As an example, during *Pre*, we first apply the preprocessing techniques (e.g., all but stemming) to the training corpus (e.g., Wikipedia), then generate word embeddings, then convert a classification dataset (e.g., IMDB) into word embedding representation, and finally classify using LSTM. Conversely, for *Post*, we first generate word embeddings from a training corpus (e.g., Wikipedia), then apply the preprocessing techniques (e.g., all but stemming) to the classification dataset (e.g., IMDB), which is then converted to word vector representation, and finally classified using LSTM [16].

The results of this experiment are presented in Table 6, where we observe that incorporating preprocessing into the training corpora before generat-

[16]Note: For settings including stemming, the classification data is also stemmed in order to obtain a compatible vocabulary.

Models	Processing	IMDB	Semeval	Airline	IAC	Onion	Reddit	Alm	ISEAR	SSEC
CBOW	Basic	84.91	56.89	68.11	69.15	71.02	63.58	45.22	59.73	55.84
	All	88.41	60.25	71.39	71.57	73.61	65.27	48.81	62.48	57.42
	All - `neg`	83.02	56.03	69.28	69.55	70.25	64.18	46.00	60.42	55.93
	All - `pos`	85.69	57.21	71.00	70.08	72.29	64.82	47.53	62.28	56.25
	All - `punc`	84.00	57.36	70.46	70.01	72.02	65.00	47.68	61.84	56.64
	All - `spell`	86.19	58.26	70.98	70.59	72.85	65.00	47.29	61.63	57.00
	All - `stop`	**91.10**	61.00	73.00	72.31	74.50	68.20	**52.39**	64.29	58.46
	All - `stem`	88.76	**62.19**	**73.25**	**72.36**	**75.69**	**68.53**	50.28	**65.33**	**59.28**
Skip-gram	Basic	84.00	55.94	68.36	69.20	71.68	63.74	45.01	59.45	55.62
	All	87.00	59.99	71.29	71.25	73.82	65.67	48.51	**65.02**	57.13
	All - `neg`	84.97	56.11	69.00	70.17	70.04	64.55	46.28	60.54	55.86
	All - `pos`	86.21	57.62	70.25	70.85	73.22	65.47	47.49	63.44	56.00
	All - `punc`	85.00	57.20	70.00	70.77	72.00	65.00	47.10	61.72	56.49
	All - `spell`	85.75	58.49	70.26	70.89	72.63	65.18	47.14	61.25	56.84
	All - `stop`	**89.76**	**61.74**	72.19	**72.00**	**75.69**	68.29	**52.01**	64.00	58.14
	All - `stem`	89.66	60.28	**73.66**	71.98	75.24	**68.72**	51.39	63.44	**59.01**
BERT	Basic	90.11	70.82	90.23	71.19	76.30	59.74	57.81	65.70	65.39
	All	91.86	71.76	91.73	73.66	78.72	62.60	59.74	67.80	67.49
	All - `neg`	90.33	70.52	91.04	72.00	77.07	61.44	58.14	66.59	66.10
	All - `pos`	91.01	71.20	91.66	73.31	78.45	62.04	59.01	66.25	68.13
	All - `punc`	91.59	71.50	91.60	73.18	78.54	62.27	59.60	67.25	67.27
	All - `spell`	91.78	71.13	91.34	73.02	78.40	62.00	59.44	67.21	67.30
	All - `stop`	**94.18**	**73.81**	**94.85**	**75.80**	**79.10**	**65.39**	**60.73**	**69.33**	**69.81**
	All - `stem`	92.19	71.94	92.03	74.49	77.93	63.74	60.16	68.00	67.05

Table 5: F-score results of evaluating the effect of preprocessing factors using different models on Wikipedia corpus. The overall best results are shown in **bold**.

Models	Processing	IMDB	Semeval	Airline	IAC	Onion	Reddit	Alm	ISEAR	SSEC
CBOW	Post	87.49	59.33	71.28	69.87	74.20	67.13	47.19	62.00	56.27
	Pre	**88.76**	62.19	**73.25**	**72.36**	**75.69**	68.53	50.28	**65.33**	*59.28*
	Both	88.10	**62.41**	73.00	71.86	75.00	**70.10**	**50.39**	64.52	58.20
Skip-gram	Post	88.14	60.41	71.85	70.22	75.07	67.00	50.44	62.08	56.00
	Pre	**89.76**	**61.74**	72.19	**72.00**	**75.69**	68.29	**52.01**	64.00	**58.14**
	Both	89.33	61.25	**73.58**	71.62	75.48	**68.74**	51.68	**65.29**	58.03
BERT	Post	94.58	70.25	92.35	74.69	77.10	63.38	58.40	68.20	67.17
	Pre	94.18	**73.81**	**94.85**	**75.80**	**79.10**	**65.39**	**60.73**	**69.33**	**69.81**
	Both	**94.63**	72.41	93.00	75.19	78.69	65.17	60.33	69.06	68.43

Table 6: F-score results of evaluating the effect of preprocessing word embeddings training corpus vs. preprocesssing evaluation datasets

ing word vectors (Pre) outperforms preprocessing classification datasets (Post) across all nine datasets of the three affective tasks. Interestingly though, preprocessing both the bodies of text (Both) appears to be of little benefit, suggesting the importance of preprocessing training corpora used for obtaining word embeddings.

5.3 Evaluating Proposed Model against State-of-the-art Baselines

While not a primary focus of this paper, in this final experiment we compare the performance of our preprocessed word embeddings against those of six state-of-the-art pretrained word embeddings[17].

(i) **GloVe**: Global vectors for word representations (Pennington et al., 2014) were trained on aggregated global word co-occurrences. We use the vectors trained on GloVe6B 6 billion words[18], uncased, from Wikipedia and Gigaword. (ii) **SSWE**: Sentiment Specific Word Embeddings (unified model)[19] were trained using a corpus of 10 million tweets to encode sentiment information into the continuous representation of words (Tang et al., 2014). (iii) **FastText**: These pretrained word vectors[20], based on sub-word character n-grams were trained on Wikipedia using fastText (Bojanowski et al., 2017), an extension of the word2vec model.

[17] These vectors obtained from their original repositories have been used without any modifications.

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

[19] http://ir.hit.edu.cn/dytang/paper/sswe/embedding-results.zip

[20] https://github.com/facebookresearch/fastText

Models	IMDB	Semeval	Airline	IAC	Onion	Reddit	Alm	ISEAR	SSEC
GloVe	85.64	70.29	70.21	70.19	71.39	63.57	56.21	65.30	58.40
SSWE	80.45	69.27	78.29	64.85	52.74	50.73	51.00	54.71	52.18
FastText	75.26	68.55	70.69	55.74	58.29	59.37	52.28	25.40	53.20
DeepMoji	69.79	62.10	71.03	65.67	70.90	53.08	46.33	58.20	58.90
EWE	71.28	60.27	67.81	67.43	70.06	55.02	58.33	66.09	58.94
Our best results:									
CBOW	91.10	62.19	73.25	72.36	75.69	68.53	52.39	65.33	59.28
Skip-gram	89.76	61.74	73.66	72.00	75.69	**68.72**	52.01	65.02	59.01
BERT	**94.18**	**73.81**	**94.85**	**75.80**	**79.10**	65.39	**60.73**	**69.33**	**69.81**

Table 7: F-score results of comparing against state-of-the-art word embeddings. The best score is highlighted in **bold**, and the second best result is underlined.

(iv) **DeepMoji**: These word embeddings[21] were trained using BiLSTM on 1.2 billion tweets with emojis (Felbo et al., 2017). (v) **EWE**: Emotion-enriched Word Embeddings[22] were learned on 200,000 Amazon product reviews corpus using an LSTM model (Agrawal et al., 2018).

From the results in Table 7, we notice that BERT is best on eight out of nine datasets except one sarcasm dataset (Reddit), while word2vec CBOW is the second best on four datasets. Overall, our analysis suggests that preprocessing at word embedding stage (Pre) works well for all the three affective tasks.

5.4 Analyzing the Three Affective Tasks

Figure 2 summarizes the results obtained for all three tasks in terms of (a) absolute F-scores and (b) relative improvement (best preprocessing over `Basic` preprocessing). The IMDB dataset achieves the highest F-score overall, most likely because it consists of movie reviews which are much longer than the text from other genres. As expected, the binary classification task of sentiment analysis and sarcasm detection achieve comparable results, while the multiclass emotion classification typically has much lower F-scores. The most interesting observation, however, is noticed in Fig. 2(b) where the emotion datasets show the highest relative improvement, indicating that multiclass classification tasks may benefit the most from applying preprocessing at word embedding stage (Pre).

6 Conclusions

We systematically examined the role of preprocessing training corpora used to induce word representations for affect analysis. While all preprocessing techniques improved performance to a certain extent, our analysis suggests that the most noticeable increase is obtained through negation processing (`neg`). The overall best performance is achieved by applying all the preprocessing techniques, except stopwords removal (All-`stop`). Interestingly, incorporating preprocessing into word representations appears to be far more beneficial than applying it in a downstream task to classification datasets. Moreover, while all the three affective tasks (sentiment analysis, sarcasm detection and emotion classification) benefit from our proposed preprocessing framework, our analysis reveals that the multiclass emotion classification task benefits the most. Exploring the space of subsets of our preprocessing factors might yield more interesting combinations; we leave this for future work.

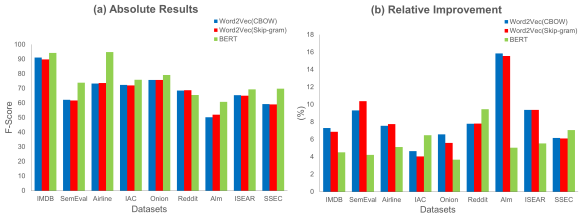

Figure 2: Absolute F-scores vs. relative improvement

Acknowledgements

We thank the anonymous reviewers for their insightful comments. This work is funded by Natural Sciences and Engineering Research Council of Canada (NSERC) and the Big Data Research, Analytics, and Information Network (BRAIN) Alliance established by the Ontario Research Fund Research Excellence Program (ORF-RE). In particular, we thank Majid Taghdimi from Questrade to provide us with the computing resources and help in the parallelization algorithm. We would also like to thank Dr. Heidar Davoudi for the helpful discussions and insights in this project.

[21] https://github.com/bfelbo/DeepMoji

[22] https://www.dropbox.com/s/wr5ovupf7yl282x/ewe_uni.txt

References

Ameeta Agrawal and Aijun An. 2012. Unsupervised emotion detection from text using semantic and syntactic relations. In *Proceedings of the The 2012 IEEE/WIC/ACM International Joint Conferences on Web Intelligence and Intelligent Agent Technology-Volume 01*, pages 346–353. IEEE Computer Society.

Ameeta Agrawal, Aijun An, and Manos Papagelis. 2018. Learning emotion-enriched word representations. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 950–961.

Giulio Angiani, Laura Ferrari, Tomaso Fontanini, Paolo Fornacciari, Eleonora Iotti, Federico Magliani, and Stefano Manicardi. 2016. A comparison between preprocessing techniques for sentiment analysis in twitter. In *Proceedings of the 2nd International Workshop on Knowledge Discovery on the WEB, KDWeb*.

Nastaran Babanejad, Ameeta Agrawal, Heidar Davoudi, Aijun An, and Manos Papagelis. 2019. Leveraging emotion features in news recommendations. In *Proceedings of the 7'th International Workshop on News Recommendation and Analytics (INRA'19) in conjunction with RecSys'19, Copenhagen, Denmark, September 16 - 20, 2019*.

Farah Benamara, Baptiste Chardon, Yannick Mathieu, Vladimir Popescu, and Nicholas Asher. 2012. How do negation and modality impact on opinions? In *Proceedings of the Workshop on Extra-Propositional Aspects of Meaning in Computational Linguistics*, ExProM '12, pages 10–18, Stroudsburg, PA, USA. Association for Computational Linguistics.

Erik Boiy, Pieter Hens, Koen Deschacht, and Marie-Francine Moens. 2007. Automatic sentiment analysis in on-line text. In *Proceedings of the 11th International Conference on Electronic Publishing ELPUB2007*.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Jose Camacho-Collados and Mohammad Taher Pilehvar. 2018. On the role of text preprocessing in neural network architectures: An evaluation study on text categorization and sentiment analysis. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*. Association for Computational Linguistics.

Ebba Cecilia and Alm Ovesdotter. 2008. *Affect in text and speech*. ProQuest, Citeseer.

Billy Chiu, Gamal Crichton, Anna Korhonen, and Sampo Pyysalo. 2016. How to train good word embeddings for biomedical nlp. In *Proceedings of the 15th workshop on biomedical natural language processing*, pages 166–174.

Taner Danisman and Adil Alpkocak. 2008. Feeler: Emotion classification of text using vector space model. In *Proceedings of the AISB 2008 Symposium on Affective Language in Human and Machine, AISB 2008 Convention Communication, Interaction and Social Intelligence*, volume 1, page 53.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Nemanja Djuric, Jing Zhou, Robin Morris, Mihajlo Grbovic, Vladan Radosavljevic, and Narayan Bhamidipati. 2015. Hate speech detection with comment embeddings. In *Proceedings of the 24th International Conference on World Wide Web*, WWW '15 Companion, page 29–30, New York, NY, USA. Association for Computing Machinery.

Manaal Faruqui, Jesse Dodge, Sujay K Jauhar, Chris Dyer, Eduard Hovy, and Noah A Smith. 2014. Retrofitting word vectors to semantic lexicons. *arXiv preprint arXiv:1411.4166*.

Bjarke Felbo, Alan Mislove, Anders Søgaard, Iyad Rahwan, and Sune Lehmann. 2017. Using millions of emoji occurrences to learn any-domain representations for detecting sentiment, emotion and sarcasm. In *Proceedings of the 2017 International Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Vachagan Gratian and Marina Haid. 2018. Braint at iest 2018: Fine-tuning multiclass perceptron for implicit emotion classification. In *Proceedings of the 9th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 243–247.

Kazuma Hashimoto, Caiming Xiong, Yoshimasa Tsuruoka, and Richard Socher. 2017. A joint many-task model: Growing a neural network for multiple NLP tasks. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1923–1933, Copenhagen, Denmark. Association for Computational Linguistics.

Daniel Hershcovich, Assaf Toledo, Alon Halfon, and Noam Slonim. 2019. Syntactic interchangeability in word embedding models. *arXiv preprint arXiv:1904.00669*.

Zhao Jianqiang and Gui Xiaolin. 2017. Comparison research on text pre-processing methods on twitter sentiment analysis. *IEEE Access*, 5:2870–2879.

Mikhail Khodak, Nikunj Saunshi, and Kiran Vodrahalli. 2017. A large self-annotated corpus for sarcasm. *arXiv preprint arXiv:1704.05579*.

Yanghoon Kim, Hwanhee Lee, and Kyomin Jung. 2018. AttnConvnet at SemEval-2018 task 1: Attention-based convolutional neural networks for multi-label emotion classification. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 141–145, New Orleans, Louisiana. Association for Computational Linguistics.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Omer Levy and Yoav Goldberg. 2014. Dependency-based word embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, volume 2, pages 302–308.

Omer Levy, Yoav Goldberg, and Ido Dagan. 2015. Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association for Computational Linguistics*, 3:211–225.

Pierre Lison and Andrey Kutuzov. 2017. Redefining context windows for word embedding models: An experimental study. In *Proceedings of the 21st Nordic Conference on Computational Linguistics (NoDaLiDa)*, pages 284–288, Gothenburg, Sweden. Association for Computational Linguistics.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA. Association for Computational Linguistics.

Oren Melamud, David McClosky, Siddharth Patwardhan, and Mohit Bansal. 2016. The role of context types and dimensionality in learning word embeddings. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1030–1040, San Diego, California. Association for Computational Linguistics.

Tomas Mikolov, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781.

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

Rishabh Misra and Prahal Arora. 2019. Sarcasm detection using hybrid neural network. *arXiv preprint arXiv:1908.07414*.

Saif M Mohammad, Parinaz Sobhani, and Svetlana Kiritchenko. 2017. Stance and sentiment in tweets. *ACM Transactions on Internet Technology (TOIT)*, 17(3):26.

Hala Mulki, Chedi Bechikh Ali, Hatem Haddad, and Ismail Babaoğlu. 2018. Tw-star at semeval-2018 task 1: Preprocessing impact on multi-label emotion classification. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 167–171.

Preslav Nakov, Alan Ritter, Sara Rosenthal, Veselin Stoyanov, and Fabrizio Sebastiani. 2016. SemEval-2016 task 4: Sentiment analysis in Twitter. In *Proceedings of the 10th International Workshop on Semantic Evaluation*, SemEval '16, San Diego, California. Association for Computational Linguistics.

Shereen Oraby, Vrindavan Harrison, Lena Reed, Ernesto Hernandez, Ellen Riloff, and Marilyn Walker. 2016. Creating and characterizing a diverse corpus of sarcasm in dialogue. In *Proceedings of the 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 31–41, Los Angeles. Association for Computational Linguistics.

Chaitali G. Patil and Sandip Patil. 2013. Use of porter stemming algorithm and svm for emotion extraction from news headlines. In *International Journal of Electronics, Communication and Soft Computing Science and Engineering*.

Samuel Pecar, Michal Farkas, Marian Simko, Peter Lacko, and Maria Bielikova. 2018. Nl-fiit at iest-2018: Emotion recognition utilizing neural networks and multi-level preprocessing. In *Proceedings of the 9th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 217–223.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

S Lovelyn Rose, R Venkatesan, Girish Pasupathy, and P Swaradh. 2018. A lexicon-based term weighting scheme for emotion identification of tweets. *International Journal of Data Analysis Techniques and Strategies*, 10(4):369–380.

Hassan Saif, Miriam Fernandez, Yulan He, and Harith Alani. 2014. On stopwords, filtering and data sparsity for sentiment analysis of twitter. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 810–817, Reykjavik, Iceland. European Language Resources Association (ELRA).

Hendrik Schuff, Jeremy Barnes, Julian Mohme, Sebastian Padó, and Roman Klinger. 2017. Annotation, modelling and analysis of fine-grained emotions on a stance and sentiment detection corpus. In *Proceedings of the 8th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 13–23.

Dibyendu Seal, Uttam K Roy, and Rohini Basak. 2020. Sentence-level emotion detection from text based

on semantic rules. In *Information and Communication Technology for Sustainable Development*, pages 423–430. Springer.

Florian Strohm. 2017. The impact of intensifiers, diminishers and negations on emotion expressions. B.S. thesis, University of Stuttgart.

Symeon Symeonidis, Dimitrios Effrosynidis, and Avi Arampatzis. 2018. A comparative evaluation of preprocessing techniques and their interactions for twitter sentiment analysis. *Expert Systems with Applications*, 110:298–310.

Duyu Tang, Furu Wei, Nan Yang, Ming Zhou, Ting Liu, and Bing Qin. 2014. Learning sentiment-specific word embedding for twitter sentiment classification. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1555–1565, Baltimore, Maryland. Association for Computational Linguistics.

Ivan Vulić, Simon Baker, Edoardo Maria Ponti, Ulla Petti, Ira Leviant, Kelly Wing, Olga Majewska, Eden Bar, Matt Malone, Thierry Poibeau, Roi Reichart, and Anna Korhonen. 2020. Multi-simlex: A large-scale evaluation of multilingual and cross-lingual lexical semantic similarity.

Harald G Wallbott and Klaus R Scherer. 1986. How universal and specific is emotional experience? evidence from 27 countries on five continents. *Information (International Social Science Council)*, 25(4):763–795.

Peng Xu, Andrea Madotto, Chien-Sheng Wu, Ji Ho Park, and Pascale Fung. 2018. Emo2Vec: Learning generalized emotion representation by multi-task training. In *Proceedings of the 9th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 292–298, Brussels, Belgium. Association for Computational Linguistics.