The Role of Preprocessing for Word Representation Learning in Affective Tasks

Nastaran Babanejad, Heidar Davoudi, Ameeta Agrawal, Aijun An, Manos Papagelis

Abstract—Affective tasks, including sentiment analysis, emotion classification, and sarcasm detection have drawn a lot of attention in recent years due to a broad range of useful applications in various domains. The main goal of affect detection tasks is to recognize *states* such as mood, sentiment, and emotions from textual data (e.g., news articles or product reviews). Despite the importance of utilizing *preprocessing* steps in different stages (i.e., word representation learning and building a classification model) of affect detection tasks, this topic has not been studied well. To that end, we explore whether applying various preprocessing methods (stemming, lemmatization, stopword removal, punctuation removal and so on) and their combinations in different stages of the affect detection pipeline can improve the model performance. The are many preprocessing approaches that can be utilized in affect detection tasks. However, their influence on the final performance depends on the type of preprocessing and the stages that they are applied. Moreover, the preprocessing impacts vary across different affective tasks. Our analysis provides thorough insights into how preprocessing steps can be applied in building an effect detection pipeline and their respective influence on performance.

Index Terms—Word Representation, Language Representation, Pretrained Language Models, Affective Tasks, Text Preprocessing, Word Embeddings, Emotion Classification, Sentiment analysis, Sarcasm Detection.

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 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, news articles, and tweets [1], [2], [3].

Most recent models of affect analysis typically employ pretrained word (or sentence) embeddings that have been obtained under the assumption of the distributional hypothesis [4], [5], [6]. 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 many natural language processing (NLP) tasks, they are known to be less suitable for affective tasks in particular [7], [8], [9]. For example, the pretrained `word2vec` embeddings [4] estimate the pair of words '*happy*' and '*sad*' to be more similar than the pair '*happy*' and '*joy*', which is counterintuitive, and might affect the performance of the models that depend on it.

To address the limitations of traditional word embeddings in affective tasks, several techniques have been proposed, including task-specific fine-tuning [5], retrofitting [10], representing emotion with vectors using a multi-task training framework [11] and generating affective word embeddings [9], [12], [13], to name a few. Another approach for improving the performance of word vectors includes optimization of preprocessing hyperparameters such as context window size and subsampling [14]. While these strategies have demonstrated evidence of improving the accuracy performance in tasks such as word similarity, word analogy, and others [15], studying their effectiveness in affective tasks has not received considerable attention and remains less explored.

Our work is motivated by the observation that pre-processing factors such as stemming, removing stopwords, and many others make up an integral part of nearly every improved text classification model, and affective systems in particular [16], [17]. However, little work has been done towards understanding the role of pre-processing techniques applied at 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 linguistically motivated pre-processing techniques (e.g., stemming, stopword removal) as applied to the training corpora before computing the text representations, and evaluated on three categories of affective tasks, including *sentiment analysis*, *emotion classification* and *sarcasm detection*. Towards that end, we systematically analyze the effectiveness of applying pre-processing to large embedding-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 pre-processing techniques earlier into word embedding models, instead of later on in a downstream classification model? *(ii)* which pre-processing techniques yield the most benefit in affective tasks? *(iii)* what is the effect of customizing or modifying the most common pre-processing methods for learning word representation learning applied to the embedding-training phase for affective tasks? *(iv)* does text pre-processing downstream classification datasets, customized for a specific affective task, provide any improvement over using text pre-processing techniques for a general affective task? *(v)* which combinations of pre-processing factors are best suited for each affective task? *(vi)* which com-

- *N. Babanejad, A. An And M. Papagelis are with the Department of Electrical and Computer Engineering, York Univesity, Toronto, Canada, ON. E-mail: nasba, aan, papaggel}@eecs.yorku.ca*
- *H. Davoudi is with Faculty of Science, Ontario Tech University, Canada. E-mail: heidar.davoudi@ontariotechu .ca*
- *A. Agrawal is with the Department of Computer Science, Portland State University, United States. E-mail: ameeta@cs.pdx.edu*

Manuscript received April 19, 2005; revised August 26, 2015.

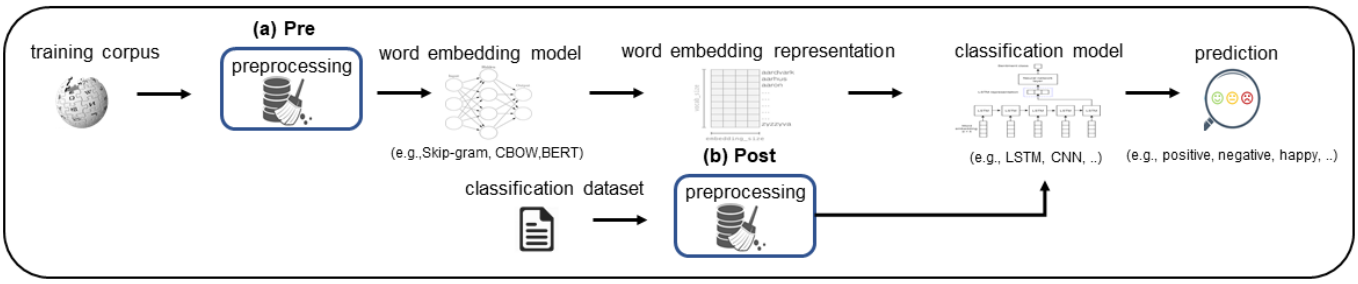

Fig. 1. Framework of applying pre-processing in different stages in the affect prediction system: (a) Pre and/or (b) Post.

binations of pre-processing techniques are more beneficial when they are applied to the embedding-training phase and/or which are more suited when applied to the classification datasets? *(vii)* does task-specific pre-processing of word embeddings provide any improvement over state-of-the-art pre-trained word embeddings, and if yes, by how much?

Figure 1 illustrates our proposed framework of applying pre-processing in different stages of developing a classification model for affect detection, where pre-processing occurs in two places: (a) at the embedding-training phase (**`Pre`**), and (b) at the classification model building phase (**`Post`**). Pre-processing techniques in (a) are applied to the embedding-training corpus of the embedding model (e.g., Wikipedia) and in (b) to the classification dataset (e.g., IMDb movie reviews).

In brief, the main contributions of our work are as follows:

- We conduct a comprehensive analysis of the role of pre-processing techniques in affective tasks (including sentiment analysis, emotion classification and sarcasm detection), employing seven different models, across nine datasets;
- We perform a comparative analysis of the performance of pre-trained word embedding models when pre-processing is applied at the embedding-training phase (i.e., training corpus) and/or at the downstream task phase (i.e., classification dataset).
- We investigate the usefulness of customizing or modifying the most common pre-processing methods for learning word representation. In particular, we propose different combinations of pre-processing factors to investigate the effectiveness of various sets of pre-processing methods in different affective tasks.
- We conduct extensive experiments which demonstrate that applying customized pre-processing on embedding-training corpora and task-specific pre-processing for each affective task on classification datasets can significantly enhance the model's performance.
- We evaluate the performance of our best pre-processed word vectors (those that are obtained by training on a corpus to which pre-processing methods have been applied) against some state-of-the-art pre-trained word embedding models.
- We make our *source code* and *data* publicly available to encourage reproducibility of the results[1].

An earlier version of this work was published in [18], where we reported our initial results on this investigation. This paper extends the investigation as follows. In this extended version, we propose additional task-specific pre-processing factors including Extended Parts-of-Speech (pos-ext), Keeping Punctuation (k-punc), Emoticons (emojis), Custom Stopwords (stop-c), and Lemmatization (lemma) and different combinations of pre-processing factors suitable for each affective task namely , Pre-processing Factors (Set A), Pre-processing Factors (Set B), Pre-processing Factors (Set C) applied either to embedding-training corpora (i.e. at the embedding-training phase) or to classification tasks (i.e. with downstream task datasets) or both. In the previous ACL version of the paper we investigated a smaller set of factors and their combinations and also the investigation was not tailored to specific affective tasks, but the general affective task. We also utilize four additional word embedding models, namely, FastText (CBOW), FastText (Skip-gram), GloVe and ELMo in this extended version. Additionally, we investigate the effects of our models on an additional Sentiment Multi-class Classification evaluation dataset (The Stanford Sentiment Treebank dataset (SST-5)) which is different from our other three traditional binary sentiment classification datasets by having five sentiment labels including very negative, negative, neutral, positive, or very positive, with score values ranging from 1 to 5 accordingly.

The rest of the paper is organized as follows: Section 2 presents an overview of the related work. Section 3 elaborates on the pre-processing techniques studied in this work. Section 4 describes the experimental evaluation framework. In Section 5, a comprehensive analysis and discussion of the results is provided. Lastly, Section 6 concludes the paper with key insights of the research.

2 Related Work

In this section, first, we provide an overview of related work on *sentiment/affect analysis*, and then present *pre-processing classification datasets* and *pre-processing embedding-training corpora*, and how our work aims to bridge the gap between those efforts.

2.1 Sentiment and Affect Analysis

Although affect analysis can be considered as a classification problem, it is a suitcase of many NLP sub-tasks. For example, Cambria et al. [19] organized the involved sub-problems into a three-layer structure (i.e., syntactic, semantics, and pragmatics) and identified 15 NLP sub-task problems. Recently, there has been a growing interest in *sarcasm* [20], *polarity* [21], and *aspect* [22] detection in pragmatics layer sub-tasks. While sarcasm detection can be seen as a classification task, it requires a deep understanding of natural language. Therefore, deep models have been used widely to extract features for sarcasm detection task [20], [23]. In sentiment/polarity analysis, some approaches focus on binary

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

classification of polarity, and others pay attention to fine-grained categorization predicting intensity of the sentiment [24].

The other important line of research is *aspect-based sentiment analysis*, where the goal is to detect the sentiment polarities (e.g., positive, negative, and neutral) of different aspects in a sentence (e.g., food, service). Recent works [25], [26] utilize Graph Neural Networks (GNNs) based on the dependency trees to capture the long-term syntactic dependencies from contextual words to aspect words. For example, Linag et al. [27] constructed a graph upon the dependency tree and affective common-sense knowledge in a sentence. The dependency graph is fed into a GNN-based model, learning a graph representation of the sentence. Laing et al. [28] argued that the dependency tree may introduce *noisy association*. To alleviate the noise effect, they proposed a graph attention network, which exploits the syntax information of the constituent tree and models the sentiment-aware context of each aspect and sentiment relations across aspects. Zhang et al. [28] argued that the general contextual embedding trained based on the next sentence prediction and masked language model can capture the semantics of the overall sentences and hardly pay attention to relevant information for the specific aspects in the sentence. Therefore, they suggest dynamic reweighting BERT [5], considering the aspect aware semantics in the building pre-train model.

Due to the finer granularity of aspect-based sentiment analysis, the *cost of annotation* and creating large-scale labelled data is much higher than conventional sentiment analysis. He et al. [29] cast aspect-based sentiment prediction into a *multi-task learning* paradigm. They considered three related sub-tasks, namely, Aspect term Extraction (AE), Opinion Extraction (OE), and aspect-level sentiment classification. They argued that using multi-task learning has some challenges, such as *imbalance* label distribution (some sub-tasks may not have enough training). Therefore, they proposed a self-training method generating *pseudo labels*, to mitigate the impacts of insufficient and imbalanced data. The pseudo-label approaches [30] trust the generated fake labels by a teacher network trained by a small number of samples and try to address the lack of enough training data. To address the noisy pseudo labels, they proposed a novel meta-weightier generating sub-task-specific weights used to evaluate the quality of the pseudo-labelled data. The model consists of a teacher, a student, and an extra meta-weighter model, which are trained jointly. In another work, Zhang et al. [31] showed that the method based on pseudo labels are limited to fit a uniform granularity situation. As such, they proposed a dual-granularity pseudo labeling framework leveraging the labels from both granularities (i.e., general sentiments, and aspect-based sentiments).

Although models based on deep neural network architectures can provide the most accurate predictions in many domains including affect analysis, they mostly lack the *interpretability* and the advantage of symbolic domain knowledge. *Neurosymbolic* frameworks aim to mitigate the issue by combining learnable neural parameters and symbolic knowledge bases. They not only can improve sentiment prediction [32], but they can increase interpretability. Li et al. [33] proposed a neurosymbolic model for Emotion Recognition in Conversation (ERC). They argued that we need to learn different types of dependencies (e.g., speaker-utterance) in a dialogue. The proposed a model fusing symbolic dependency knowledge, concept-level commonsense, and sentiment knowledge. They exploited relational graph convolutional networks, relation-aware concept representations, and convolutional self-attention techniques for this purpose. Their experiment results show the state-of-the-art performance on three benchmark datasets. In another work, Camberia et al. [34] proposed a common sense-based neuro-symbolic framework for sentiment analysis. They used subsymbolic techniques such as auto-regressive language models as well as kernel approaches to build a hierarchical common-sense knowledge graph. The main idea is to translate an input sentence into a sentence expressed by a set of primitives in the knowledge graph, and then connect these primitives to their corresponding emotions and polarity labels. The proposed approach predicts the polarity on the fly upon the building blocks of meaning in comparison to classic approaches associating polarity to a static list of affect words. They showed that the suggested model is more interpretable, trustworthy, and explainable while having comparable accuracy to other state-of-the-art models.

2.2 Pre-processing Classification Datasets (`Post`)

Pre-processing is a vital step in natural language processing and therefore, evaluation of pre-processing techniques has long been a part of many NLP systems in general, and affective models in particular. [35] indicated that, despite its popular use in Twitter sentiment analysis, the use of pre-compiled stoplist has a negative impact on the classification performance. [36] analyzed various pre-processing methods such as stopwords removal, stemming, negation, emoticons, and so on, and found stemming to be most effective for sentiment analysis. Similarly, [37] found that lemmatization increases accuracy. [38] 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.

Pre-processing techniques such as punctuation and negation [39] or pos-tagging and negation [40] make up a common component of many emotion classification models [17], [41]. One of the earliest works [16] 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 $\rightarrow$ NOThappy). Although stemming may conflate the affective connotation of some words, it has been shown to improve classification accuracy overall [16], [42]. Negations have also been found to be beneficial, whereas considering intensifiers and diminishers did not lead to any improvements [43].

[44] also highlight the importance of pre-processing when using user-generated content, with emoticons processing being the most effective. Along the same lines, while [45] found pos-tags to be useful, [46] ignored pos-tagging because of its effect of reducing the classification accuracy. In the task of emotion classification, [47] examined the role of four pre-processing 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.

The aforementioned works describe pre-processing techniques as applied directly to the classification datasets in affective systems, i.e., **`Post`**. In contrast, we examine the effectiveness of directly incorporating these known effective pre-processing techniques further "upstream" into the embedding-training corpus, i.e., **`Pre`**.

2.3 Pre-processing Embedding-Training Corpora (`Pre`)

Through a series of extensive experiments, particularly those related to context window size and dimensionality, [14] 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 pre-processing settings. [15] 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 pre-processing factors such as context window size [48], [49], dimensionality [49], syntactic dependencies [50], [51], [52] and their effect on NLP tasks including word similarity [48], tagging, parsing, relatedness, and entailment [53] and biomedical [54] 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 pre-processing factors of generating word embeddings specifically tailored to affective tasks, which have received little attention.

A recent study which investigated the role of tokenizing, lemmatizing, lowercasing and multiword grouping [55] as applied to UMBC WebBase training corpus and evaluated on sentiment analysis found simple tokenization to be generally adequate.

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

2.4 Task-specific Pre-processing

Sentiment Analysis. Haddi et al., [56] investigated the role of different combination of text pre-processing methods (e.g., HTML tags removal, non-alphabetic signs removal, whitespace removal, POS-tagging, abbreviation expansion, stemming, stopwords removal, spell correction, and negation handling) in sentiment analysis and found that reducing noise through pre-processing improved the performance of the classification task. Carrillo et al., [57] proposed a model of negation, intensifiers, and modality especially conceived for sentiment analysis tasks by handling negation with an antonym dictionary which replaces the to-be-negated word with its antonym (e.g., "not good" replaced with "bad"). Furthermore, Jianqiang and Xiaolin [58] showed that the most effective pre-processing methods for sentiment classification include replacing negative mentions (e.g., "won't" into "will not"), removing URL links, spelling correction, removing punctuation, removing stopwords, and stemming.

Sarcasm Detection. The results in [59], [60], [61] demonstrate the importance of sarcasm indicators. For instance, Burgers et al., [60] introduced a set of sarcasm indicators such as emoticons (i.e., ":p", ":)"), spelling correction, interjections (i.e., "ah", "hmm"), exclamation marks (e.g., "!", "?"), quotation marks, and intensifiers words (e.g., "too", "greatest", "best", "really") that explicitly signal if an utterance is sarcastic. POS-tagging, stemming [62], [63] and lemmatization [64], [65] have also been found useful for sarcasm detection. However, most of these indicators will be removed by applying the pre-processing factors such as punctuation removal, stopwords removal and retaining selective POS tagged words (i.e. by retrieving only four classes) [18].

Emotion Detection. In the task of emotion classification, Agrawal and An [66] observed that stemming and POS-tagging of the text improved the accuracy. Goddar et al., [67] indicated that interjections such as *"yay"* and *"wow"* can be useful indicators of different emotions and help in improving the classification performance. Furthermore, Mulik et al., [47] found stemming, lemmatization and emoji tagging to be the most effective factors. Similarly, other studies have also assessed the effects of intensifiers, interjections [68], [69], POS-tagging, negation, punctuation [70], and emoticons [69] in emotion detection.

3 Proposed Methodology

This section describes various combination of pre-processing factors for embedding-training corpus and the classification datasets, as well as the suggested order in which to apply these pre-processing factors.

We experiment with three sets of pre-processing factors namely (`Set A`), (`Set B`), and (`Set C`). Details of each set of pre-processing factors are described in the following sub-sections.

3.1 Pre-processing Factors (`Set A`)

In this section we describe the first set of pre-processing techniques called (`Set A`), which are the most common and known effective pre-processing techniques based on prior research as applied either directly to classification datasets in affective systems or incorporating such pre-processing further "upstream" into the embedding-training corpus. In addition to the most common pre-processing methods, we also consider negation since negation is a mechanism that can transform a positive argument into its inverse rejection [71] and improper processing of negation in sentences may lead to misclassification of affect in affective systems. Note that in `Set A` pre-processing we apply the same pre-processing factors at different stages of the affective systems.

The details of each pre-processing technique used in this set are as follows:

Basic: A group of common text pre-processing applied at the very beginning, such as removing HTML tags, removing numbers, and lowercasing.

Punctuation Removal (`punc`): 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 them as they are, assuming that they represent natural language text and its associated complexities. In this step, we perform spell checking[3] to identify misspelled or unknown words and generate a list of suggested replacements for an identified misspelled or unknown word. If there is only one option for correction, we use it; otherwise, we rank the multiple suggested replacements according to their frequency in the ukWac corpora[4], and select the most frequent one for replacement.

Negation (`neg`): Negation is a mechanism that transforms a positive argument into its inverse rejection [71]. 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:

2. https://www.nltk.org/_modules/nltk/tokenize/regexp.html
3. https://pypi.org/project/pyspellchecker/
4. https://www.sketchengine.eu/ukwac-british-english-corpus/

(i) Compilation of an antonym dictionary: The first stage involves compiling an antonym dictionary using the WordNet corpus [72]. For every synset, there are three possibilities: finding no antonym, one antonym or multiple antonyms. The first two cases are trivial involving unambiguous replacements. In the case of the third option (ambiguous replacement), which also appears to be the most common case, amongst the many choices available, we consider the antonym with the highest 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., the negated word) is extracted and its antonym looked up in the antonym dictionary. If an antonym is found, the negation word as well as the negated word are replaced with it.

As an illustration, consider 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 our antonym dictionary (i.e., '*sad*') and finally, 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., the tags corresponding to NN*, VB*, JJ*, 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 pre-processing technique in NLP tasks. We use the NLTK Snowball stemmer for stemming the text.

3.2 Pre-processing Factors (`Set B`)

In this section we describe the second set of pre-processing techniques called (`Set B`) as applied either directly to classification datasets in affective systems or incorporated further "upstream" into the embedding-training corpus. Note that in (`Set B`) we also apply the same pre-processing factors at different stages of the affective systems.

It is possible that not all of the most common pre-processing methods in (`Set A`) are equally effective in all three affective tasks. For instance, some might even harm the classification results according to previous studies, such as stemming and stopwords removal when applied at the embedding-training phase [18]. In addition, others [73], [74] demonstrated that the contribution of use/non-use of stopwords depends on the corpus they are applied to for a downstream task and that for some corpora, various pre-processing methods might be even irrelevant such as removing punctuation or stopwords. One of the possible explanations for the phenomenon that stopword removal or stemming at training phase on embedding-training corpora may harm the performance results in affective tasks is that while applying them at the classification datasets can help clean the dataset and reduce the noises, but not applying these two pre-processing factors "upstream" into the embedding-training corpus may help the word embedding models to better understand the context. Moreover, previous studies demonstrated the effectiveness of affect indicators such as interjections (i.e.,"ah", "hmm"), exclamation marks (e.g., "!", "?"), quotation marks, emoticons (i.e., ":p", ":)"), and intensifiers words (e.g., "too", "greatest", "best", "really") in affective tasks [59], [60], [63], [64], [65], [75]. However, most of these indicators will be removed by applying pre-processing factors such as punctuation removal, stopwords removal and POS tagging (i.e. by retrieving only four classes which we considered in the previous section (`Set A`). Towards that end to systematically analyze the effectiveness of different combination of pre-processing techniques in our three affective tasks, we modify some of the pre-processing factors in (`Set A`) and also add new pre-processing methods which potentially can be more effective in affective tasks to build a new set that is called (`Set B`) for our experiment.

Details of each pre-processing technique used in this set is as follows:

Basic , **Negation (`neg`)** and **Spellcheck (`spell`)**: We keep these pre-processing factors the same as in the previous section from (`Set A`).
Extended Parts-of-Speech (`pos-ext`): In addition to the four classes mentioned earlier, i.e., nouns, verbs, adjectives, and adverbs, interjections have also served as useful features, especially in affective tasks. Thus, using the NLTK pos-tagger, for each sentence in the corpus we extend the set of POS-tags that we retain, i.e., NN*, VB*, JJ*, RB*, as well as UH*.
Keeping Punctuation (`k-punc`): Since some studies indicate the benefits of punctuation in affective tasks [76], [77], we keep all the special characters, exclamation marks and punctuation. During tokenization, we make sure there are spaces between words and punctuation to keep the punctuation in separate tokens. Moreover, in the case of consecutive punctuation marks (e.g. !!!, :-)), we keep all of them in the same token.
Emoticons (`emojis`): The abundance of emoticons and emojis in user-generated social media posts and reviews provides evidence of their propensity for expressing a range of different emotions, and they have been proven to be very effective in various affective tasks [78], [79]. Therefore, we convert the emoticons and emojis into text (e.g., converting 😫 to "tired face" and 😏 to "smirking face" or ;) to "wink face") using the Demoji module[6], a library for converting emojis into associated text strings.

3.3 Pre-processing Factors (`Set C`)

While in the previous sets, (`Set A`) and (`Set B`), we considered pre-processing techniques as applied either directly to classification datasets in affective systems or incorporated these pre-processing techniques further "upstream" into the embedding-training corpus, in this section we focus on creating a number of pre-processing factors that can be applied to classification datasets in affective systems. Based on the insights derived from prior literature, in this section, we propose a number of task-specific pre-processing factors called (`Set C`) to be applied to classification datasets in addition to the pre-processing techniques described earlier. Some illustrative examples of task-specific pre-processing are shown in Table 1. Note, in (`Set C`) we apply different pre-processing factors for each affective task directly to classification

5. https://pypi.org/project/negspacy/

6. https://pypi.org/project/demoji/

TABLE 1
Example of case studies of different pre-processing in affective tasks.

Affective Task	Pre-Processing	Example
Sentiment Analysis	Negation Correction	*I won't walk again.* → *I will not walk again.*
	Negation	*I feel not good.* → *I feel bad.*
	Intensifiers	*This work is extremely hard.*
	Interjections	*Wow, Is this your house?.*
Sarcasm Detection	Emoticons	*Awesome failure!* 😐
	Keeping Punctuation	*Time for you medication or mine ??!!*
	Intensifiers	*Sarcasm detection is too easy!*
	Interjections	*Oh, I'm nicer in sleep.*
Emotion Detection	Emoticons	*Are you serious now??* 😠
	Keeping Punctuation	*We are not friends anymore???!!!*
	Interjections	*Yay! We are going out tonight.*
	Intensifiers	*This is the best experience I've ever had.*

datasets and we apply pre-processing techniques in (`Set B`) as described earlier further "upstream" into the embedding-training corpus[7].

Details of additional pre-processing techniques used in this set is as follows:

Basic, **Negation (`neg`)**, **Spellcheck (`spell`)**, **Extended Parts-of-Speech** (`pos-ext`), **Keeping Punctuation** (`k-punc`) and **Emoticons** (`emojis`): we keep these pre-processing factors the same as in the previous section from (`Set B`).

Custom Stopwords (`stop-c`): Sometimes there are benefits to keeping stopwords, especially when removing them will exclude the intensifiers (e.g. "*too*", "*really*") which are helpful indicators of affect as indicated by prior literature. Therefore, we experiment with further variations of stopwords removal including either not removing the stopwords at all or employing a custom stopwords list which helps us retain the intensifier words in text[8].

Lemmatization (`lemma`): An alternative to stemming is the process of lemmatization, a morphogically-informed process of removing inflectional endings in order to reduce a word to its lemma. This may not only help to reduce the noise in text due to variability but also provide a more abstract affective meaning of a sentence. We lemmatize the text using NLTK Wordnet Lemmatizer[9]. Before finding the correct lemma, part-of-speech tagging must be performed. For instance, "*working*", *"worked"*, and *"works"* are all lemmatized to *"work"* in a verb form. When a certain word can belong to more than one grammatical category, depending on its part-of-speech tag within the sentence (e.g., noun or verb), all the related lemmas are kept.

Inspired by prior literature with some illustrative examples in Table 1 we create a number of task-specific pre-processing in this set for each affective task. As it was mentioned earlier, in contrast to pre-processing factors in (`Set A`) and (`Set B`) that apply the same pre-processing factor at different stages of the affective systems regardless of the downstream task, in this section we conduct our experiments by applying pre-processing factors in (`Set B`) to embedding-training corpora and the following task-specific sequences of pre-processing to classification datasets for each affective task:

Sentiment Analysis: `Basic`, `punc`, `spell`, `neg`, `pos-ext`, `c-stop`, and `stem`.

Sarcasm Detection: `Basic`, `spell`, `neg`, `pos-ext`, `k-punc`, `emojis`, `lemma`, or `stem`.

Emotion Detection: `Basic`, `spell`, `neg`, `pos-ext`, `k-punc`, `emojis`, `lemma`, or `stem`.

Note, for sarcasm detection and emotion detection datasets we apply lemmatization and stemming separately and we report the best results.

Sequence of Pre-processing Factors: While some pre-processing 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 pre-processing factors: spellchecking, negation handling, pos classes, removing stopwords, lemmatization/stemming.

4 Experimental Setup

In this section, we present the embedding-training corpora, the word embedding models, the affective evaluation datasets, and the evaluation classification set up used in our experiments.

4.1 Embedding-Training Corpora

For evaluating the effects of the proposed pre-processing factors, we consider two types of plain text corpora originating from two different domains, including,

(i) **News**: This corpus consists of 142,546 news articles sourced from 15 American publications, spanning from 2013 to early 2018[10].

(ii) **Wikipedia**: Comparatively a much larger corpus than the News, this corpus consists of 23,046,187 articles from English Wikipedia [11].

The details of the two corpora are summarized in Table 2 including their vocabulary and corpus sizes before (i.e., Basic) and after applying all the various pre-processing settings. Although,

7. For settings which include stemming/lemmatization, the embedding-training corpora is also stemmed/lemma in order to obtain a compatible vocabulary.

8. List of intensifier words are collected from: https://github.com/wyounas/homer and [80]

9. https://www.nltk.org/_modules/nltk/stem/wordnet.html

10. https://www.kaggle.com/snapcrack/all-the-news

11. https://www.kaggle.com/jkkphys/english-wikipedia-articles-20170820-sqlite

TABLE 2
Details of the embedding-training corpora (for 'Vocab' and 'Corpus', the 'size' indicates the size of the vocabulary and the total number of tokens in the corpus, respectively. The % column includes the reduction in the vocabulary and corpus size after applying the various pre-processing factors as compared to the corpus).

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

as expected, the corpus size reduces after applying any pre-processing factor, it is especially worth noting that for certain pre-processing such as POS (`pos`) and stopwords removal (`stop`), a dramatic reduction in corpus size is observed (as indicated by the % ratio of pre-processed `pos` or `stop` to Basic listed under 'Corpus'), *without any significant loss in vocabulary*. Such a reduction in corpus size inherently implies a non-trivial effect with regards to training time, a phenomenon we plan to explore in more detail in our future work.

4.2 Word Embedding Models

We obtain our pre-processed word representations by training seven different word embedding models from scratch. These include,

(i) **`word2vec` CBOW (Continuous Bag-of-Words)**, and (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 [4]. We train both these models on both News and Wikipedia embedding-training corpora. The minimum number of words is set to 5 for News and 100 for Wikipedia with window sizes set to 5 and 10, respectively, and dimensionality set to 300.

(iii) **FastText CBOW**, and (iv) **Skip-gram**: FastText is another word embedding method that is an extension of the word2vec model [81]. Instead of learning vectors for words directly, FastText represents each word based on sub-word character *n*-grams. The settings for training these models are the same as those for word2vec models, with the size of *n*-gram set to 3.

(v) **GloVe**: Global vectors for word representations [82] is a global log-bilinear regression model for the unsupervised learning of word representations where it creates a global co-occurrence matrix by estimating the probability of a given word that will co-occur with other words. We train the model with the same parameters as the original paper via adaptive gradient descent (AdaGrad), setting dimensionality to 300, number of epochs to 100 and batch size to 2048.

(vi) **ELMo**: ELMo (Embeddings from Language Models) is a character-based bidirectional contextual language model [83], where the embedding for a given word might vary based on the different contexts in which the word appears. The hyperparameters for training this model are the same as the original model, with the LSTM dimensionality of 4096, 10 epochs and number of negative samples batch of 8192.

(vii) **BERT**: BERT is another bidirectional contextualized model using the transformer architecture for learning unsupervised pre-trained language representations [5]. We train the model using the BERT large uncased architecture (24-layer, 1024-hidden, 16-heads, 340M parameters) with the same settings for parameters as found in the original paper.

We train these models 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[12] 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

To extrinsically evaluate the quality of our pre-trained language representations, we conduct a series of experiments on a number of datasets for three downstream affective tasks, namely sentiment analysis, emotion classification and sarcasm detection. Some illustrative examples of text are shown in Table 3 while Table 4 presents the details of all the evaluation datasets.

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[13] includes 50,000 movie reviews for sentiment analysis, consisting of 25,000 negative and 25,000 positive reviews [84]. (ii) **Semeval 2016**: This sentiment analysis in Twitter dataset[14] consists of 14,157 tweets where 10,076 of them are positive and 4,081 negative [85]. (iii) **Airline**: This sentiment analysis dataset[15] consists of 11,541 tweets about six U.S. airlines from February 2015, with 9,178 tweets labeled as positive and 2,363 negative.

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) **IAC**: A subset of the Internet Argument Corpus [86], this dataset contains response utterances annotated for sarcasm. We extract 3260 instances from the general sarcasm type.[16]. (ii) **Onion**: This news headlines dataset [17] collected sarcastic versions of current events from The Onion and non-sarcastic news headlines from HuffPost [87], resulting in a total 28,619

12. https://docs.microsoft.com/en-us/cognitive-toolkit/multiple-gpus-and-machines
13. http://ai.stanford.edu/ amaas/data/sentiment/
14. http://alt.qcri.org/semeval2016/task4/index.php
15. https://www.kaggle.com/crowdflower/twitter-airline-sentiment
16. https://nlds.soe.ucsc.edu/sarcasm2
17. https://github.com/rishabhmisra/News-Headlines-Dataset-For-Sarcasm-Detection

TABLE 3
Some examples of text instances from the evaluation datasets.

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
· *Been saying that ever since the first time I heard about creationsism*	not-sarcastic	IAC
· *ford develops new suv that runs purely on gasoline*	sarcastic	Onion
· *Remember, it's never a girl's fault, it's always the man's fault.*	sarcastic	Reddit
· *The ladies danced and clapped their hands for joy.*	happy	Alm
· *At work, when an elderly man complained unjustifiably about me and distrusted me.*	anger	ISEAR
· *if this heat is killing me i don't wanna know what the poor polar bears are going through*	sadness	SSEC

TABLE 4
Details of the evaluation datasets.

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

records. (iii) **Reddit**: Self-Annotated Reddit Corpus (SARC)[18] 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 [88].

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) **Alm**: This dataset contains sentences from fairy tales marked with one of five emotion categories: angry-disgusted, fearful, happy, sad and surprised [89]. (ii) **ISEAR**: This dataset contains narratives of personal experiences evoking emotions [90]. We use a subset of the data consisting of five categories: sadness, anger, disgust, fear, joy. (iii) **SSEC**: The Stance Sentiment Emotion Corpus [91] is the re-annotation of the SemEval 2016 Twitter stance and sentiment corpus [92] with emotion labels including anger, joy, sadness, fear, surprise. [19].

4.4 Evaluation Classification Setup

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:

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

18. SARC v0.0: https://nlp.cs.princeton.edu/SARC/0.0/
19. SSEC: http://www.romanklinger.de/ssec/

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

The optimizer is Adam [93], 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

In this section, we analyze the impact of different sets of pre-processing techniques in word representation learning on affect analysis.

5.1 Effect of Pre-processing Factors (`Set A`)

A primary goal of this work is to identify the most effective set of pre-processing factors for training word embeddings that can be useful for affective tasks. Table 5 details the results of our experiments comparing the performance of individual pre-processing factors in (`Set A`) as well as those of ablation studies (i.e., including all the factors but one) on News corpus. To see whether the best method significantly improves the **Basic** method, a paired t-test is conducted, whose p-value is shown beside the best method in the first column. The t-test result indicates that there is a significant difference between the results of the best method and those of the **Basic** method.

Observing the performance of the individual factors on the News corpus, we note that even a single simple pre-processing technique can bring improvements, thereby validating our intuition of incorporating pre-processing into embedding-training corpora of word representations. Second, negation (`neg`) processing appears to be consistently the most 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 POS classes. On the other hand, removing stopwords (`stop`), spellchecking (`spell`) and stemming (`stem`) yield little improvement and mixed results. Interestingly, applying all the pre-processing 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 pre-processing factors except stemming

TABLE 5
Evaluating the effect of pre-processing (Set A) using CBOW and Skip-gram word embedding models trained on the News corpus. The overall best results in terms of F-score are in **bold**. The best result among the ones under one pre-processing setting is underlined. The p-value from a paired t-test is shown beside the best method, which compares the results from the best method and the ones from the **Basic** method. Both t-tests show a significant difference between the two methods.

Training Models	Processing	IMDB	Semeval	Airline	IAC	Onion	Reddit	Alm	ISEAR	SSEC
Word2Vec (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
($p < 0.0005$)	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**
Word2Vec (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
($p < 0.0001$)	All - stem	88.42	60.67	**67.39**	**69.08**	72.00	**62.36**	37.44	**59.48**	**55.23**

(All-stem). Moreover, when the models are trained on a different corpus, i.e., Wikipedia, the results of which are summarized in Table 6, in all but one case (GloVe being the outlier), we note that the best performance is obtained by combining all the pre-processing factors except stemming (All-stem) or except removing stopwords (All-stop). In addition, we need to note that FastText works well with rare words. Thus, even if a word was not seen during training, it can be broken down into n-grams to get its embeddings. However, ELMo is character-based, which means that the model does not have a pre-defined vocabulary of words used for training, but it rather extracts the word embeddings from the constituent characters of the words. These differences can make the word embedding model more robust and generalizable to downstream tasks. Moreover, the p-values from the paired t-test shown in Table 6 indicate that the best method is significantly better than the **Basic** method in terms of F-score. In addition, 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 Application of Pre-processing Factors in (Set A) on Embedding-Training Corpora (Pre) vs. on Downstream Classification Dataset (Post)

We investigate the difference between applying pre-processing methods in (Set A) to the embedding-training corpora for generating word embeddings **Pre** and applying the same pre-processing methods in this set to the classification datasets **Post**. As an example, during **Pre**, we first apply the pre-processing techniques (e.g., all but stemming) to the embedding-training corpus (e.g., Wikipedia), 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 embedding-training corpus (e.g., Wikipedia), then apply the pre-processing 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 [20].

The results of this experiment are presented in Table 7, where we observe that incorporating pre-processing techniques into the embedding-training corpora before generating word vectors (**Pre**) outperforms the technique where pre-processing is only applied to the classification datasets (**Post**) across all nine datasets of the three affective tasks. Interestingly though, pre-processing both the bodies of text (**Both**) appears to be of little benefit using this set of pre-processing methods, suggesting the importance of pre-processing embedding-training corpora used for obtaining word embeddings using the pre-processing methods in (Set A). In addition, as shown in Table 7, there is a significant difference between the averages of **Pre** and **Post** for six out of seven models, while there is a significantly large difference between the averages of **Pre** and **Both** for four models. Figure 2 summarizes the results obtained for all three tasks in terms of (a) absolute F-scores and (b) relative improvement (best pre-processing methods in set A over Basic pre-processing). 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

20. For settings which include stemming, the classification data is also stemmed in order to obtain a compatible vocabulary.

TABLE 6
Evaluating the effect of pre-processing (`Set A`) using different models trained on the Wikipedia corpus. The p-value from the paired t-test is shown in the first column beside the best method, which compares the results from the best method and those from the **Basic** method.

Models	Processing	IMDB	Semeval	Airline	IAC	Onion	Reddit	Alm	ISEAR	SSEC
	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
Word2Vec (CBOW)	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
($p < 0.0001$)	All - `stem`	88.76	**62.19**	**73.25**	**72.36**	**75.69**	**68.53**	50.28	**65.33**	**59.28**
	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
Word2Vec (Skip-gram)	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
($p < 0.0005$)	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**
	Basic	74.68	68.20	70.29	57.83	60.50	60.13	48.29	26.07	51.47
	All	77.12	69.86	71.69	60.69	63.39	63.07	50.77	28.42	53.89
	All - `neg`	75.01	68.90	70.83	58.81	61.25	61.74	49.21	27.89	52.04
FastText (CBOW)	All - `pos`	78.51	69.17	70.26	60.57	61.94	62.29	49.86	28.06	52.71
	All - `punc`	76.92	69.37	71.14	60.19	62.71	62.78	50.44	28.30	53.65
	All - `spell`	76.85	69.73	71.00	59.90	62.18	62.41	50.04	28.00	53.65
	All - `stop`	**80.37**	**71.08**	72.39	**62.74**	64.79	64.33	**53.37**	30.24	55.28
($p < 0.0005$)	All - `stem`	79.45	70.10	**73.06**	61.83	**65.48**	**65.51**	52.19	**30.61**	**55.74**
	Basic	75.00	68.41	70.41	58.13	61.12	60.72	49.13	26.68	52.07
	All	78.30	69.73	71.65	61.52	64.57	63.61	51.03	28.76	54.21
	All - `neg`	75.86	68.59	70.75	59.33	62.03	61.58	50.00	28.04	52.84
FastText (Skip-gram)	All - `pos`	79.24	70.33	71.00	60.00	63.08	62.11	50.41	29.47	53.07
	All - `punc`	78.01	69.51	71.10	60.94	64.00	62.84	50.94	28.01	53.67
	All - `spell`	77.90	69.50	71.25	61.11	64.27	63.02	50.79	28.63	53.91
($p < 0.0001$)	All - `stop`	**81.83**	**71.30**	**73.29**	62.81	**66.12**	65.71	**53.69**	**30.48**	**56.32**
	All - `stem`	80.82	70.82	72.61	**63.28**	65.75	**66.24**	52.76	30.07	56.15
	Basic	83.51	69.12	70.01	69.48	70.21	62.76	54.31	64.77	56.33
($p < 0.0001$)	All	**87.32**	**73.22**	**74.39**	**73.14**	**74.19**	**67.29**	58.51	**67.34**	**59.70**
	All - `neg`	84.06	70.09	71.37	71.15	71.39	63.24	55.10	65.31	57.63
GloVe	All - `pos`	85.33	72.76	72.19	72.35	72.06	65.07	56.33	62.82	58.12
	All - `punc`	86.72	71.47	72.77	72.62	73.95	65.88	57.90	66.74	59.37
	All - `spell`	86.48	71.28	73.61	72.69	73.70	65.79	58.04	66.70	58.42
	All - `stop`	86.39	72.84	73.64	72.84	73.67	66.19	57.61	67.13	59.10
	All - `stem`	86.25	73.00	73.41	72.33	73.61	66.25	**58.93**	66.05	59.33
	Basic	86.34	69.47	82.11	70.18	71.65	65.48	60.24	65.00	66.20
	All	88.63	71.80	83.91	72.61	73.30	66.93	63.20	67.58	69.45
	All - `neg`	87.00	70.21	82.79	71.58	72.00	66.10	61.06	66.10	67.24
ELMo	All - `pos`	87.64	70.68	83.00	72.00	72.49	66.47	61.70	65.72	67.81
	All - `punc`	88.41	71.67	83.27	72.40	73.17	66.71	62.47	67.00	68.73
	All - `spell`	88.23	71.55	83.16	72.33	73.09	66.50	62.59	67.11	68.10
($p < 0.0001$)	All - `stop`	**90.27**	**73.54**	**85.61**	**74.04**	**74.45**	**68.74**	64.17	**69.28**	**71.01**
	All - `stem`	89.45	72.30	85.02	73.10	74.20	67.59	**64.78**	68.52	70.33
	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
BERT	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
($p < 0.0001$)	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

analysis and sarcasm detection achieve comparable results, while the multi-class 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 pre-processing at word embedding stage **`Pre`**. These observations motivate us to explore the idea of using different sets of pre-processing methods as described in the next subsections.

5.3 Effects of Pre-processing Factors in (`Set B`)

Now we turn our attention to the effects of pre-processing factors in (`Set B`) for training word embeddings for affective tasks. Table 8 details the results of our experiments comparing

TABLE 7
Evaluating the effect of pre-processing (Set A) at embedding-training corpus (Pre) vs. at classification datasets (Post) and (Both) (F-score). The t-test results (p-values) indicate the statistically-significant difference between the averages of Pre and Post (first p-value), and those of Pre and Both (second p-value).

Embedding-Training Corpus	Processing	IMDB	Semeval	Airline	IAC	Onion	Reddit	Alm	ISEAR	SSEC
($p < 0.0001; p > 0.5$) Word2Vec (CBOW)	Pre	**88.76**	62.19	**73.25**	**72.36**	**75.69**	68.53	50.28	**65.33**	**59.28**
	Post	87.49	59.33	71.28	69.87	74.20	67.13	47.19	62.00	56.27
	Both	88.10	**62.41**	73.00	71.86	75.00	**70.10**	**50.39**	64.52	58.20
($p < 0.0001; p > 0.5$) Word2Vec (Skip-gram)	Pre	**89.76**	**61.74**	72.19	**72.00**	**75.69**	68.29	**52.01**	64.00	**58.14**
	Post	88.14	60.41	71.85	70.22	75.07	67.00	50.44	62.08	56.00
	Both	89.33	61.25	**73.58**	71.62	75.48	**68.74**	51.68	**65.29**	58.03
($p < 0.0005; p > 0.5$) FastText (CBOW)	Pre	79.45	70.10	**73.06**	**61.83**	**65.48**	**65.51**	52.19	30.61	**55.74**
	Post	76.48	69.25	70.15	57.38	62.48	63.71	51.47	29.35	53.84
	Both	**79.72**	**70.54**	72.26	61.00	65.27	65.20	**52.36**	**30.65**	55.49
($p > 0.05; p < 0.01$) FastText (Skip-gram)	Pre	**81.83**	**71.30**	**73.29**	62.81	66.12	**65.71**	**53.69**	**30.48**	**56.32**
	Post	80.01	76.16	71.40	58.70	64.22	63.76	51.49	29.74	54.38
	Both	80.52	70.40	72.58	**63.02**	**66.57**	65.00	53.18	30.24	55.29
($p < 0.0005; p < 0.001$) GloVe	Pre	**87.32**	**73.22**	**74.39**	**73.14**	**74.19**	67.29	58.51	**67.34**	**59.70**
	Post	86.37	71.20	72.30	72.15	72.47	65.73	57.19	66.31	58.64
	Both	87.00	72.48	74.18	73.01	73.61	**67.32**	**58.66**	67.29	59.14
($p < 0.0005; p < 0.01$) ELMo	Pre	**90.27**	**73.54**	**85.61**	**74.04**	**74.45**	**68.74**	64.17	**69.28**	**71.01**
	Post	88.13	71.76	83.10	72.28	73.55	66.79	63.80	68.20	70.18
	Both	90.14	72.57	85.00	73.61	74.20	68.07	**64.39**	68.79	70.83
($p < 0.0001; p < 0.001$) BERT	Pre	94.18	**73.81**	**94.85**	**75.80**	**79.10**	**65.39**	**60.73**	**69.33**	**69.81**
	Post	94.58	70.25	92.35	74.69	77.10	63.38	58.40	68.20	67.17
	Both	**94.63**	72.41	93.00	75.19	78.69	65.17	60.33	69.06	68.43

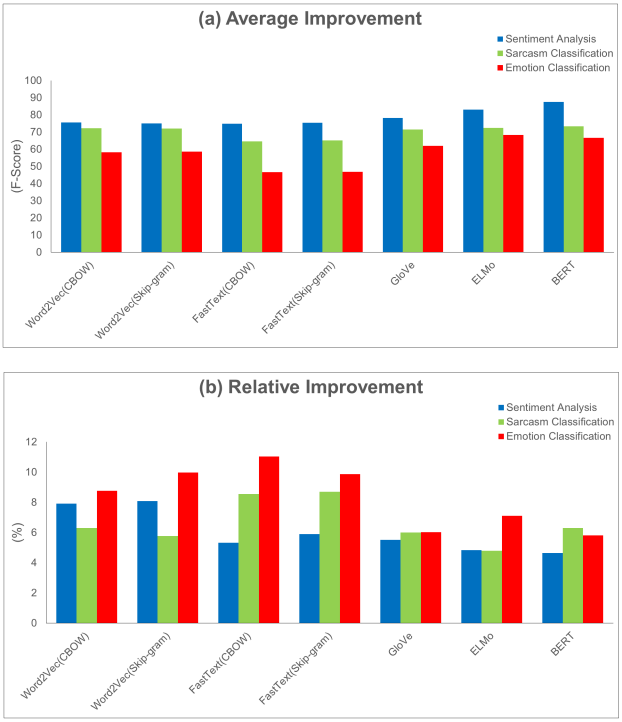

Fig. 2. Average F-scores vs. relative improvement

the performance of applying all pre-processing factors (All) in (Set A) on Wikipedia corpus and the best results of ablation studies (e.g., All−stem using pre-processing methods in (Set A) as compared to the pre-processing methods in (Set B) at embedding-training phase for all the seven models. Observing the performance of the pre-processing factors in (Set B) when we apply them at the embedding-training phase, we note that the pre-processing techniques in this set outperform the combination of pre-processing (e.g. All, or All−stem) from (Set A) in most settings consistently, thereby supporting our intuition that customizing pre-processing methods and adding new pre-processing factors that can be more beneficial for certain affective tasks can bring additional gains over combination of general pre-processing factors such as in (Set A).

5.4 Evaluating Application of Pre-processing Factors in (Set B) on Embedding-Training Corpora (Pre) vs. on Downstream Classification Dataset (Post)

In this experiment we compare the performance of models when applying pre-processing methods in (Set B) to the embedding-training corpora for generating word embeddings **Pre** and applying the same pre-processing methods to the classification datasets **Post**. The results of these experiments are presented in Table 9, where we observe similar behavior as in (Set A) that incorporating pre-processing into the embedding-training corpora before generating word vectors (Pre) outperforms pre-processing classification datasets (Post) in majority of the cases across all nine datasets of the three affective tasks. Interestingly, this time customized pre-processing both the bodies of text (Both) using pre-processing methods in this set appears to be more beneficial (as second best results) compared to pre-processing both the bodies of text using pre-processing methods in (Set A), suggesting that an appropriate combination of pre-processing methods more customized for affective tasks achieves better results than using only general and most common pre-processing factors.

5.5 Evaluating Application of Pre-processing Factors in (Set B) on Embedding-Training Corpora (Pre) vs. Pre-processing Factors in (Set C) on Downstream Classification Dataset (Post)

This experiment investigates the effect of applying pre-processing factors in (Set B) on embedding-training corpora and a number of task-specific pre-processing factors called (Set C) to be

TABLE 8
Comparing the effect of pre-processing using `Set B` with `Set A` for training word embeddings (Pre) (F-score).

Model	Processing	IMDB	Semeval	Airline	IAC	Onion	Reddit	Alm	ISEAR	SSEC
Word2Vec (CBOW)	(Set A) All	88.41	60.25	71.39	71.57	73.61	65.27	48.81	62.48	57.42
	(Set A) All - stem	88.76	62.19	73.25	72.36	75.69	68.53	50.28	65.33	59.28
	(Set B)	**90.67**	**62.74**	**74.33**	**73.08**	**76.52**	**69.15**	**53.18**	**66.19**	**60.51**
Word2Vec (Skip-gram)	(Set A) All	87.00	59.99	71.29	71.25	73.82	65.67	48.51	**65.02**	57.13
	(Set A) All - stop	89.76	61.74	72.19	72.00	75.69	68.29	52.01	64.00	58.14
	(Set B)	**89.91**	**62.73**	**73.69**	**72.85**	**76.31**	**69.24**	**52.84**	64.80	**59.28**
FastText (CBOW)	(Set A) All	77.12	69.86	71.69	60.69	63.39	63.07	50.77	28.42	53.89
	(Set A) All - stem	79.45	70.10	73.06	61.83	65.48	65.51	52.19	30.61	55.74
	(Set B)	**80.71**	**71.90**	**73.70**	**63.17**	**66.24**	**66.71**	**53.00**	**33.25**	**56.49**
FastText (Skip-gram)	(Set A) All	78.30	69.73	71.65	61.52	64.57	63.61	51.03	28.76	54.21
	(Set A) All - stop	81.83	71.30	73.29	62.81	66.12	65.71	53.69	30.48	56.32
	(Set B)	**82.93**	**72.00**	**74.15**	**63.57**	**66.80**	**66.79**	**55.38**	**32.29**	**56.63**
GloVe	(Set A) All	**87.32**	73.22	**74.39**	73.14	74.19	67.29	58.51	**67.34**	59.70
	(Set B)	86.73	**73.41**	74.00	**74.23**	**74.27**	**68.40**	**59.80**	66.85	**60.11**
ELMo	(Set A) All	88.63	71.80	83.91	72.61	73.30	66.93	63.20	67.58	69.45
	(Set A) All - stop	90.27	**73.54**	**85.61**	**74.04**	74.45	68.74	64.17	69.28	71.01
	(Set B)	**90.40**	73.20	85.03	71.19	**75.27**	**69.87**	**65.38**	**69.81**	**71.80**
BERT	(Set A) All	91.86	71.76	91.73	73.66	78.72	62.60	59.74	67.80	67.49
	(Set A) All - stop	**94.18**	73.81	94.85	75.80	79.10	65.39	60.73	69.33	69.81
	(Set B)	93.67	**74.00**	**94.88**	**79.00**	**79.84**	**66.00**	**61.18**	**70.28**	**70.33**

applied to classification datasets. In (`Set C`) we apply different pre-processing factors for each affective task (explained in Section 3.3) directly to classification datasets and we apply pre-processing techniques in (`Set B`) as described earlier (explained in Section 3.2) further "upstream" into the embedding-training corpus [21]. As an example, during **`Pre`**, for all three affective tasks, we first apply the pre-processing techniques in (`Set B`) explained in Section 3.2 to the embedding-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 (using Basic pre-processing) from a embedding-training corpus (e.g., Wikipedia), then apply the pre-processing techniques used in (`Set C`) for sentiment analysis to all the three classification datasets in sentiment analysis datasets (e.g., IMDB, Semeval, and Airline), which is then converted to word vector representation, and finally classified using LSTM.

The results of these experiments are presented in Table 10, where we observe that pre-processing both the bodies of text (Both) by applying customized pre-processing on embedding-training corpora in (`Set B`) to generate the embedding and applying task-specific pre-processing methods from (`Set C`) on each affective task dataset consistently achieves best results across all nine datasets of the three affective tasks. This observation supports our hypothesis that customized pre-processing of embedding-training corpora and task-specific pre-processing of downstream affective classification datasets may bring additional gains over task-agnostic and generic pre-processing.

5.6 Sentiment Multi-class Classification

In this experiment we investigate how our proposed approaches perform on a multi-class sentiment classification dataset called the *Stanford Sentiment Treebank* (SST5) dataset [94]. This is a collection of 11,855 sentences extracted from movie reviews and converted to parse trees with 215,154 different phrases. The (root) sentences are divided into training (8,544), validation (1,101), and testing (2,210). For the fine-grained (root) sentence classification task (SST-5), we need to predict one of the five classes (negative, weakly negative, neutral, weakly positive, or positive; scores range from 1 to 5 accordingly) for each (root) sentence in the test set.

We first investigate the difference between applying the pre-processing methods in (`Set A`) to the embedding-training corpora for generating word embeddings (**`Pre`**) and applying the same pre-processing methods in this set to the classification datasets (**`Post`**), similar to our experiments in section 5.2. The results of this experiment are presented in Table 12, where we observe that incorporating pre-processing in this set into the embedding-training corpora before generating word vectors (**`Pre`**) outperforms pre-processing classification datasets (**`Post`**) in four out of seven models. Interestingly, pre-processing both bodies of text (**`Both`**) appears to be beneficial using the pre-processing methods in (`Set A`) for three models, namely, Word2Vec (CBOW), FastText (CBOW), and ELMo. This suggests that continuous Bag-of-Words models and a character-based bidirectional contextual language model will perform better using this set of pre-processing methods for **`Both`**.

Table 13 shows the results of our experiments comparing the performance of applying all pre-processing factors (`All`) in (`Set A`), the best results of ablation studies using pre-processing factors in (`Set A`), and embedding-training using factors in (`Set B`) for all the seven models on Wikipedia corpus. Observing the performance of the pre-processing factors in (`Set B`) applied in embedding-training phase, we note that the pre-processing techniques in this set outperform the combination of pre-processing (e.g., `All`, or `All-stem`) using (`Set A`) in most cases consistently. This shows that customizing pre-processing methods and adding new pre-processing factors are more beneficial to affective tasks than general pre-processing factors in (`Set A`).

21. For settings which include stemming/lemmatization, the embedding-training corpora is also stemmed/lemmatized in order to obtain a compatible vocabulary.

TABLE 9
Evaluating the effect of pre-processing (Set B) at embedding-training corpus (Pre) vs. at classification datasets (Post) and (Both) (F-score)

Embedding-Training Corpus	Processing	IMDB	Semeval	Airline	IAC	Onion	Reddit	Alm	ISEAR	SSEC
Word2Vec (CBOW)	Pre	**90.67**	**62.74**	**74.33**	**73.08**	**76.52**	**69.15**	**53.18**	**66.19**	**60.51**
	Post	87.30	60.04	72.20	68.27	73.61	66.80	48.25	61.29	55.00
	Both	88.13	61.70	72.69	70.08	74.12	68.48	50.23	65.37	58.00
Word2Vec (Skip-gram)	Pre	**89.91**	**62.73**	**73.69**	**72.85**	**76.31**	**69.24**	**52.84**	**64.80**	**59.28**
	Post	88.01	60.22	70.25	71.13	74.28	67.45	50.62	62.00	55.70
	Both	88.57	61.85	73.20	71.08	75.00	69.00	50.74	63.12	57.21
FastText (CBOW)	Pre	**80.71**	**71.90**	**73.70**	**63.17**	**66.24**	**66.71**	**53.00**	**33.25**	**56.49**
	Post	77.30	70.10	71.27	56.80	65.30	63.78	52.67	30.27	54.18
	Both	78.69	71.25	71.69	61.38	65.84	64.37	52.73	32.80	54.80
FastText (Skip-gram)	Pre	**82.93**	**72.00**	**74.15**	**63.57**	**66.80**	**66.79**	**55.38**	**32.29**	**56.63**
	Post	78.20	67.84	70.33	59.67	63.80	61.30	51.27	30.69	54.70
	Both	79.06	70.60	73.12	62.81	65.30	64.80	54.25	30.39	55.00
GloVe	Pre	86.73	**73.41**	**74.00**	**74.23**	**74.27**	**68.40**	**59.80**	**66.85**	**60.11**
	Post	85.12	70.00	71.45	72.64	71.69	65.10	56.48	64.23	57.29
	Both	**87.29**	73.00	73.14	73.45	73.59	67.48	58.29	66.70	58.76
ELMo	Pre	**90.40**	**73.20**	**85.03**	71.19	**75.27**	**69.87**	**65.38**	**69.81**	**71.80**
	Post	86.25	70.33	82.20	69.48	73.02	66.40	63.14	67.40	69.28
	Both	90.33	72.60	83.20	**72.68**	74.11	68.00	64.21	67.62	70.37
BERT	Pre	93.67	**74.00**	**94.88**	**79.00**	**79.84**	**66.00**	**61.18**	**70.28**	**70.33**
	Post	91.83	70.12	92.00	74.04	76.81	62.71	58.02	67.90	66.80
	Both	**94.03**	72.19	92.20	76.39	77.19	63.77	60.03	68.34	67.61

Now, we compare the performance of applying pre-processing methods in (Set B) to the embedding-training corpus (**Pre**) and applying the same pre-processing methods in this set to the SST-5 classification dataset (**Post**). The results of these experiments are presented in Table 14, where we observe similar behavior as in (Set A). That is, incorporating pre-processing into the embedding-training corpus before generating word vectors (**Pre**) outperforms pre-processing classification datasets (**Post**) in majority of the cases. Moreover, customized pre-processing both the bodies of text (**Both**) using pre-processing methods in (Set B) appears to be more beneficial (as first best results in 2 models and second best results in 3 models) compared to pre-processing both the bodies of text using pre-processing methods in (Set A), suggesting that an appropriate combination of pre-processing methods customized for affective tasks achieves better results than using only general and most common pre-processing factors.

Finally, we investigate the effect of applying pre-processing factors in (Set B) on embedding-training corpora and then applying the pre-processing techniques in (Set C) for sentiment analysis in SST-5 dataset. The results of these experiments are presented in Table 15, where we observe that pre-processing both the bodies of text (**Both**) by applying customized pre-processing on embedding-training corpora in (Set B) to generate the embeddings and applying task-specific pre-processing methods in (Set C) to the sentiment analysis dataset (SST-5) consistently achieves best results across all the seven models. This observation supports our hypothesis that customized pre-processing of embedding-training corpora and task-specific pre-processing downstream affective tasks can bring additional gains over task-agnostic and generic pre-processing.

5.7 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 pre-processed word embeddings against those of three state-of-the-art pre-trained word embeddings that have been specifically trained for affective tasks [22].

(i) **SSWE**: Sentiment Specific Word Embeddings (unified model)[23] were trained using a corpus of 10 million tweets to encode sentiment information into the continuous representation of words [7].

(ii) **DeepMoji**: These word embeddings[24] were trained using BiLSTM on 1.2 billion tweets with emojis [12].

(iii) **EWE**: Emotion-enriched Word Embeddings[25] were learned using 200,000 Amazon product reviews corpus using an LSTM model [9].

From the results in Table 11, we notice that BERT is best in seven out of nine datasets except one sarcasm dataset (Reddit), and one emotion detection dataset (Alm), while ELMo generates the second best results on five datasets and one best result on emotion detection dataset (Alm). Overall, our analysis suggests that task-specific pre-processing at each word embedding stage (Both) works best for all the three affective tasks.

6 Conclusions

We systematically examined the effect of pre-processing embedding-training corpora used to induce word representations for affect analysis. While all the pre-processing techniques in the first set of pre-processing methods (Set A) improved the performance to a certain extent, our analysis suggests that the most noticeable increase is obtained through negation processing (neg). The overall best performance using combination of pre-processing factors in this set is achieved by applying all the pre-processing techniques, except stopwords removal (All-stop or stem). See section 3.3 for some insights. Interestingly, we observe in this set that incorporating pre-processing into word representations

22. These vectors were obtained from their original repositories and have been used without any modifications.
23. http://ir.hit.edu.cn/dytang/paper/sswe/embedding-results.zip
24. https://github.com/bfelbo/DeepMoji
25. https://www.dropbox.com/s/wr5ovupf7yl282x/ewe_uni.txt

TABLE 10
Evaluating the effect of pre-processing (`Set B`) at embedding-training corpus (Pre) vs. pre-processing (`Set C`) at classification datasets (Post), and (Both) (F-score)

Embedding-Training Corpus	Processing	IMDB	Semeval	Airline	IAC	Onion	Reddit	Alm	ISEAR	SSEC
Word2Vec (CBOW)	Pre	90.67	62.74	74.33	73.08	76.52	69.15	53.18	66.19	60.51
	Post	88.52	60.47	73.40	71.25	75.63	68.05	50.44	64.20	59.31
	Both	**90.81**	**63.30**	**75.07**	**74.69**	**77.80**	**70.51**	**54.20**	**67.48**	**61.02**
Word2Vec (Skip-gram)	Pre	89.91	62.73	73.69	72.85	76.31	69.24	52.84	64.80	59.28
	Post	89.10	62.03	72.45	71.62	75.69	68.14	51.66	62.70	58.00
	Both	**90.40**	**64.20**	**75.37**	**74.28**	**77.82**	**71.49**	**54.09**	**66.00**	**60.58**
FastText (CBOW)	Pre	80.71	71.90	73.70	63.17	66.24	66.71	53.00	33.25	56.49
	Post	78.29	70.18	71.02	60.39	66.18	65.01	53.00	33.28	55.70
	Both	**81.60**	**72.50**	**75.06**	**65.86**	**68.21**	**69.17**	**55.48**	**36.45**	**58.71**
FastText (Skip-gram)	Pre	82.93	72.00	74.15	63.57	66.80	66.79	55.38	32.29	56.63
	Post	80.83	69.38	72.65	62.30	65.30	64.27	55.18	31.40	55.80
	Both	**83.60**	**73.41**	**75.33**	**65.39**	**68.42**	**68.70**	**57.04**	**35.20**	**58.00**
GloVe	Pre	86.73	73.41	74.00	74.23	74.27	68.40	59.80	66.85	60.11
	Post	86.20	72.00	73.10	73.00	73.81	66.80	58.30	65.10	58.26
	Both	**87.23**	**75.08**	**75.14**	**74.40**	**76.31**	**70.25**	**61.40**	**68.71**	**62.30**
ELMo	Pre	90.40	73.20	85.03	71.19	75.27	69.87	65.38	69.81	71.80
	Post	88.67	72.80	84.61	70.39	74.69	68.80	64.20	68.07	70.30
	Both	**91.20**	**74.83**	**86.67**	**73.30**	**77.00**	**71.37**	**67.49**	**71.25**	**72.20**
BERT	Pre	93.67	74.00	94.88	79.00	79.84	66.00	61.18	70.28	70.33
	Post	93.10	73.24	92.60	75.00	78.20	64.80	59.34	69.18	69.52
	Both	**94.22**	**75.20**	**94.88**	**80.21**	**80.34**	**67.41**	**63.10**	**72.66**	**72.80**

TABLE 11
Comparing against state-of-the-art word embeddings. The best score is highlighted in **bold**, the second best result is underlined (F-score).

Models	IMDB	Semeval	Airline	IAC	Onion	Reddit	Alm	ISEAR	SSEC
SSWE	80.45	69.27	78.29	64.85	52.74	50.73	51.00	54.71	52.18
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:									
FastText (CBOW)	81.60	72.50	75.06	65.86	68.21	69.17	55.48	36.45	58.71
FastText (Skip-gram)	83.60	73.41	75.33	65.39	68.42	68.70	57.04	35.20	58.00
Word2Vec (CBOW)	90.81	63.30	75.07	74.69	77.80	70.51	54.20	67.48	61.02
Word2Vec (Skip-gram)	90.40	64.20	73.37	74.28	77.82	**71.49**	54.09	66.00	60.58
GloVe	87.23	75.08	75.14	74.40	76.31	70.25	61.40	68.71	62.30
ELMo	91.20	74.83	86.67	73.30	77.00	71.37	**67.49**	71.25	72.20
BERT	**94.22**	**75.20**	**94.88**	**80.21**	**80.34**	67.41	63.10	**72.66**	**72.80**

appears to be far more beneficial than applying it in a downstream task to classification datasets. The average F-scores across all the datasets and all the embedding-training corpora (Pre: 68.79, Post: 67.08, Both: 68.49), show a slight advantage afforded by the **`Pre`** strategy. Moreover, while all the three affective tasks (sentiment analysis, sarcasm detection and emotion classification) benefit from our proposed pre-processing methods in (`Set A`), our analysis reveals that the multi-class emotion classification task benefits the most.

Further, we examined the role of pre-processing methods in (`Set B`) by modifying and customizing the methods in (`Set A`) for affective tasks and applying them to the embedding-training corpora as well as classification datasets. Our analysis reveals that an appropriate combination of pre-processing methods more customized for affective tasks achieves better results than using only general and most common combination of pre-processing factors. We also observed that pre-processing both the bodies of text (Both) by applying customized pre-processing on embedding-training corpora in (`Set B`) to generate the embeddings and applying task-specific pre-processing methods from (`Set C`) on each affective task dataset consistently achieves best results across all the nine datasets of the three affective tasks. This observation supports our hypothesis that customized pre-processing of embedding-training corpora and task-specific pre-processing of downstream affective datasets may bring additional gains over task-agnostic and generic pre-processing. To supplement our analysis, we also compared the performance of our pre-processed word embeddings against those of three state-of-the-art pre-trained word embeddings that have been specifically trained for affective tasks. The results provided evidence that BERT is best on seven out of nine datasets except one sarcasm dataset (Reddit), and one emotion detection dataset (Alm), while ELMo is the second best on five datasets and best on one emotion detection dataset (Alm). Overall, our analysis suggests that task-specific pre-processing at each word embedding stage (Both) works best for all the three affective tasks. Finally, one interesting future work item is to investigate the effect of prepossessing when sub-word embedding models, such as the one used in [6], [95]. Another research direction is to deal with noisy environments such as social networks. Such an environment requires handling slang, abbreviations, and emerging language patterns through dynamic language modeling and domain-specific knowledge incorporation .

TABLE 12
Evaluating the effect of pre-processing (Set A) at embedding-training corpus (Pre) vs. at classification datasets (Post) and (Both) for Stanford Sentiment Treebank dataset (F-score)

Embedding-Training Corpus	Processing	SST-5
Word2Vec (CBOW)	Basic	49.01
	Pre	50.25
	Post	49.70
	Both	**51.11**
Word2Vec (Skip-gram)	Basic	50.03
	Pre	**53.34**
	Post	51.20
	Both	52.01
FastText (CBOW)	Basic	43.25
	Pre	44.93
	Post	44.00
	Both	**45.82**
FastText (Skip-gram)	Basic	40.71
	Pre	**44.28**
	Post	43.47
	Both	42.86
GloVe	Basic	51.20
	Pre	**54.07**
	Post	53.07
	Both	53.49
ELMo	Basic	51.39
	Pre	53.19
	Post	52.78
	Both	**53.67**
BERT	Basic	57.06
	Pre	**59.61**
	Post	57.50
	Both	58.14

TABLE 13
Comparing the effect of pre-processing using (Set B) with (Set A) for training word embeddings (Pre) using Stanford Sentiment Treebank dataset (F-score).

Embedding-Training Corpus	Processing	SST-5
Word2Vec (CBOW)	(Set A) All	49.30
	(Set A) All - stem	50.25
	(Set B)	**52.06**
Word2Vec (Skip-gram)	(Set A) All	51.08
	(Set A) All - stop	53.34
	(Set B)	**55.00**
FastText (CBOW)	(Set A) All	43.41
	(Set A) All - stem	44.93
	(Set B)	**45.19**
FastText (Skip-gram)	(Set A) All	41.63
	(Set A) All - stop	44.28
	(Set B)	**45.08**
GloVe	(Set A) All	54.07
	(Set B)	**56.68**
ELMo	(Set A) All	52.74
	(Set A) All - stop	**53.19**
	(Set B)	52.81
BERT	(Set A) All	57.89
	(Set A) All - stop	59.61
	(Set B)	**60.57**

TABLE 14
Evaluating the effect of pre-processing (Set B) at embedding-training corpus (Pre) vs. at classification datasets (Post) and (Both) for Stanford Sentiment Treebank dataset (F-score).

Embedding-Training Corpus	Processing	SST-5
Word2Vec (CBOW)	Pre	**52.06**
	Post	50.04
	Both	51.13
Word2Vec (Skip-gram)	Pre	**55.00**
	Post	52.18
	Both	54.71
FastText (CBOW)	Pre	**45.19**
	Post	44.23
	Both	43.12
FastText (Skip-gram)	Pre	**45.08**
	Post	44.17
	Both	43.80
GloVe	Pre	**56.68**
	Post	53.82
	Both	54.27
ELMo	Pre	52.81
	Post	51.65
	Both	**53.04**
BERT	Pre	60.57
	Post	59.27
	Both	**61.02**

TABLE 15
Evaluating the effect of pre-processing (Set B) at embedding-training corpus (Pre) vs. pre-processing (Set C) at classification datasets (Post), and (Both) for Stanford Sentiment Treebank dataset (F-score)

Embedding-Training Corpus	Processing	SST-5
Word2Vec (CBOW)	Pre	52.06
	Post	51.33
	Both	**53.09**
Word2Vec (Skip-gram)	Pre	55.00
	Post	53.90
	Both	**55.89**
FastText (CBOW)	Pre	45.19
	Post	45.13
	Both	**47.25**
FastText (Skip-gram)	Pre	45.08
	Post	44.61
	Both	**46.17**
GloVe	Pre	56.68
	Post	55.11
	Both	**56.83**
ELMo	Pre	52.81
	Post	51.60
	Both	**53.05**
BERT	Pre	60.57
	Post	61.79
	Both	**62.19**

REFERENCES

[1] N. Djuric, J. Zhou, R. Morris, M. Grbovic, V. Radosavljevic, and N. Bhamidipati, "Hate speech detection with comment embeddings," in *Proceedings of the 24th International Conference on World Wide Web*, ser. WWW '15 Companion. New York, NY, USA: Association for Computing Machinery, 2015, p. 29–30. [Online]. Available: https://doi.org/10.1145/2740908.2742760

[2] N. Babanejad, A. Agrawal, H. Davoudi, A. An, and M. Papagelis, "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.*, 2019.

[3] R. Satapathy, C. Guerreiro, I. Chaturvedi, and E. Cambria, "Phonetic-based microtext normalization for twitter sentiment analysis," in *2017 IEEE International Conference on Data Mining Workshops (ICDMW)*, 2017, pp. 407–413.

[4] T. Mikolov, K. Chen, G. S. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *CoRR*, vol. abs/1301.3781, 2013.

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

[6] L. Zhuang, L. Wayne, S. Ya, and Z. Jun, "A robustly optimized BERT pre-training approach with post-training," in *Proceedings of the 20th Chinese National Conference on Computational Linguistics*. Huhhot, China: Chinese Information Processing Society of China, Aug. 2021, pp. 1218–1227. [Online]. Available: https://aclanthology.org/2021.ccl-1.108

[7] D. Tang, F. Wei, N. Yang, M. Zhou, T. Liu, and B. Qin, "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)*. Baltimore, Maryland: Association for Computational Linguistics, Jun. 2014, pp. 1555–1565. [Online]. Available: https://www.aclweb.org/anthology/P14-1146

[8] A. Agrawal and A. An, "Selective co-occurrences for word-emotion association," in *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, 2016, pp. 1579–1590.

[9] A. Agrawal, A. An, and M. Papagelis, "Learning emotion-enriched word representations," in *Proceedings of the 27th International Conference on Computational Linguistics*, 2018, pp. 950–961.

[10] M. Faruqui, J. Dodge, S. K. Jauhar, C. Dyer, E. Hovy, and N. A. Smith, "Retrofitting word vectors to semantic lexicons," *arXiv preprint arXiv:1411.4166*, 2014.

[11] P. Xu, A. Madotto, C.-S. Wu, J. H. Park, and P. Fung, "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*. Brussels, Belgium: Association for Computational Linguistics, Oct. 2018, pp. 292–298. [Online]. Available: https://www.aclweb.org/anthology/W18-6243

[12] B. Felbo, A. Mislove, A. Søgaard, I. Rahwan, and S. Lehmann, "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)*, 2017.

[13] A. Agrawal and A. An, "Affective representations for sarcasm detection," in *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, 2018, pp. 1029–1032.

[14] O. Levy, Y. Goldberg, and I. Dagan, "Improving distributional similarity with lessons learned from word embeddings," *Transactions of the Association for Computational Linguistics*, vol. 3, pp. 211–225, 2015.

[15] P. Lison and A. Kutuzov, "Redefining context windows for word embedding models: An experimental study," in *Proceedings of the 21st Nordic Conference on Computational Linguistics*. Gothenburg, Sweden: Association for Computational Linguistics, May 2017, pp. 284–288. [Online]. Available: https://www.aclweb.org/anthology/W17-0239

[16] T. Danisman and A. Alpkocak, "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*, vol. 1, 2008, p. 53.

[17] C. G. Patil and S. Patil, "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*, 2013.

[18] N. Babanejad, A. Agrawal, A. An, and M. Papagelis, "A comprehensive analysis of preprocessing for word representation learning in affective tasks," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, Jul. 2020, pp. 5799–5810. [Online]. Available: https://www.aclweb.org/anthology/2020.acl-main.514

[19] E. Cambria, S. Poria, A. Gelbukh, and M. Thelwall, "Sentiment analysis is a big suitcase," *IEEE Intelligent Systems*, vol. 32, no. 6, pp. 74–80, 2017.

[20] N. Babanejad, H. Davoudi, A. An, and M. Papagelis, "Affective and contextual embedding for sarcasm detection," in *Proceedings of the 28th international conference on computational linguistics*, 2020, pp. 225–243.

[21] M. Wankhade, A. C. S. Rao, and C. Kulkarni, "A survey on sentiment analysis methods, applications, and challenges," *Artificial Intelligence Review*, pp. 1–50, 2022.

[22] W. Zhang, X. Li, Y. Deng, L. Bing, and W. Lam, "A survey on aspect-based sentiment analysis: Tasks, methods, and challenges," *arXiv preprint arXiv:2203.01054*, 2022.

[23] J. Plepi and L. Flek, "Perceived and intended sarcasm detection with graph attention networks," in *Findings of the Association for Computational Linguistics: EMNLP 2021*. Punta Cana, Dominican Republic: Association for Computational Linguistics, Nov. 2021, pp. 4746–4753. [Online]. Available: https://aclanthology.org/2021.findings-emnlp.408

[24] M. Thelwall, *The Heart and Soul of the Web? Sentiment Strength Detection in the Social Web with SentiStrength*. Cham: Springer International Publishing, 2017, pp. 119–134. [Online]. Available: https://doi.org/10.1007/978-3-319-43639-5_7

[25] K. Sun, R. Zhang, S. Mensah, Y. Mao, and X. Liu, "Aspect-level sentiment analysis via convolution over dependency tree," in *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*, 2019, pp. 5679–5688.

[26] C. Zhang, Q. Li, and D. Song, "Aspect-based sentiment classification with aspect-specific graph convolutional networks," *arXiv preprint arXiv:1909.03477*, 2019.

[27] B. Liang, H. Su, L. Gui, E. Cambria, and R. Xu, "Aspect-based sentiment analysis via affective knowledge enhanced graph convolutional networks," *Knowledge-Based Systems*, vol. 235, p. 107643, 2022.

[28] S. Liang, W. Wei, X.-L. Mao, F. Wang, and Z. He, "Bisyn-gat+: Bi-syntax aware graph attention network for aspect-based sentiment analysis," in *Findings of the Association for Computational Linguistics: ACL 2022*, 2022, pp. 1835–1848.

[29] K. He, R. Mao, T. Gong, C. Li, and E. Cambria, "Meta-based self-training and re-weighting for aspect-based sentiment analysis," *IEEE Transactions on Affective Computing*, 2022.

[30] D. Mallis, E. Sanchez, M. Bell, and G. Tzimiropoulos, "Unsupervised learning of object landmarks via self-training correspondence," *Advances in Neural Information Processing Systems*, vol. 33, pp. 4709–4720, 2020.

[31] Y. Zhang, M. Zhang, S. Wu, and J. Zhao, "Towards unifying the label space for aspect-and sentence-based sentiment analysis," in *Findings of the Association for Computational Linguistics: ACL 2022*, 2022, pp. 20–30.

[32] J. Kocoń, J. Baran, M. Gruza, A. Janz, M. Kajstura, P. Kazienko, W. Korczyński, P. Miłkowski, M. Piasecki, and J. Szołomicka, "Neuro-symbolic models for sentiment analysis," in *International Conference on Computational Science*. Springer, 2022, pp. 667–681.

[33] W. Li, L. Zhu, R. Mao, and E. Cambria, "Skier: A symbolic knowledge integrated model for conversational emotion recognition," in *AAAI (to appear)*, 2023.

[34] E. Cambria, Q. Liu, S. Decherchi, F. Xing, and K. Kwok, "Senticnet 7: a commonsense-based neurosymbolic ai framework for explainable sentiment analysis," *Proceedings of LREC 2022*, 2022.

[35] H. Saif, M. Fernandez, Y. He, and H. Alani, "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)*. Reykjavik, Iceland: European Language Resources Association (ELRA), may 2014, pp. 810–817.

[36] G. Angiani, L. Ferrari, T. Fontanini, P. Fornacciari, E. Iotti, F. Magliani, and S. Manicardi, "A comparison between preprocessing techniques for sentiment analysis in twitter," in *Proceedings of the 2nd International Workshop on Knowledge Discovery on the WEB, KDWeb*, 2016.

[37] S. Symeonidis, D. Effrosynidis, and A. Arampatzis, "A comparative evaluation of pre-processing techniques and their interactions for twitter sentiment analysis," *Expert Systems with Applications*, vol. 110, pp. 298–310, 2018.

[38] Z. Jianqiang and G. Xiaolin, "Comparison research on text pre-processing methods on twitter sentiment analysis," *IEEE Access*, vol. 5, pp. 2870–2879, 2017.

[39] S. L. Rose, R. Venkatesan, G. Pasupathy, and P. Swaradh, "A lexicon-based term weighting scheme for emotion identification of tweets," *International Journal of Data Analysis Techniques and Strategies*, vol. 10, no. 4, pp. 369–380, 2018.

[40] D. Seal, U. K. Roy, and R. Basak, "Sentence-level emotion detection from text based on semantic rules," in *Information and Communication Technology for Sustainable Development*. Springer, 2020, pp. 423–430.

[41] Y. Kim, H. Lee, and K. Jung, "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*. New Orleans, Louisiana: Association for Computational Linguistics, Jun. 2018, pp. 141–145.

[42] A. Agrawal and A. An, "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*. IEEE Computer Society, 2012, pp. 346–353.

[43] F. Strohm, "The impact of intensifiers, diminishers and negations on emotion expressions," B.S. thesis, University of Stuttgart, 2017.

[44] S. Pecar, M. Farkas, M. Simko, P. Lacko, and M. Bielikova, "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*, 2018, pp. 217–223.

[45] V. Gratian and M. Haid, "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*, 2018, pp. 243–247.

[46] E. Boiy, P. Hens, K. Deschacht, and M.-F. Moens, "Automatic sentiment analysis in on-line text." in *Proceedings of the 11th International Conference on Electronic Publishing ELPUB2007*, 2007.

[47] H. Mulki, C. B. Ali, H. Haddad, and I. Babaoğlu, "Tw-star at semeval-2018 task 1: Preprocessing impact on multi-label emotion classification," in *Proceedings of The 12th International Workshop on Semantic Evaluation*, 2018, pp. 167–171.

[48] D. Hershcovich, A. Toledo, A. Halfon, and N. Slonim, "Syntactic interchangeability in word embedding models," *arXiv preprint arXiv:1904.00669*, 2019.

[49] O. Melamud, D. McClosky, S. Patwardhan, and M. Bansal, "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*. San Diego, California: Association for Computational Linguistics, jun 2016, pp. 1030–1040.

[50] O. Levy and Y. Goldberg, "Dependency-based word embeddings," in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, vol. 2, 2014, pp. 302–308.

[51] I. Vulić, S. Baker, E. M. Ponti, U. Petti, I. Leviant, K. Wing, O. Majewska, E. Bar, M. Malone, T. Poibeau, R. Reichart, and A. Korhonen, "Multi-simlex: A large-scale evaluation of multilingual and cross-lingual lexical semantic similarity," 2020.

[52] D. Chen and C. Manning, "A fast and accurate dependency parser using neural networks," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 740–750. [Online]. Available: https://aclanthology.org/D14-1082

[53] K. Hashimoto, C. Xiong, Y. Tsuruoka, and R. Socher, "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)*. Copenhagen, Denmark: Association for Computational Linguistics, Sep. 2017, pp. 1923–1933. [Online]. Available: https://www.aclweb.org/anthology/D17-1206

[54] B. Chiu, G. Crichton, A. Korhonen, and S. Pyysalo, "How to train good word embeddings for biomedical nlp," in *Proceedings of the 15th workshop on biomedical natural language processing*, 2016, pp. 166–174.

[55] J. Camacho-Collados and M. T. Pilehvar, "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, 2018. [Online]. Available: http://dx.doi.org/10.18653/v1/w18-5406

[56] E. Haddi, X. Liu, and Y. Shi, "The role of text pre-processing in sentiment analysis," *Procedia Computer Science*, vol. 17, p. 26–32, 12 2013.

[57] J. Carrillo-de Albornoz and L. Plaza, "An emotion-based model of negation, intensifiers, and modality for polarity and intensity classification," *Journal of the American Society for Information Science and Technology*, vol. 64, pp. 1618–1633, 08 2013.

[58] Z. Jian-qiang and G. Xiaolin, "Comparison research on text pre-processing methods on twitter sentiment analysis," *IEEE Access*, vol. 5, pp. 2870–2879, 2017.

[59] D. Ghosh, A. Richard Fabbri, and S. Muresan, "The Role of Conversation Context for Sarcasm Detection in Online Interactions," in *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*. Saarbrücken, Germany: Association for Computational Linguistics, 2017, pp. 186–196. [Online]. Available: http://aclweb.org/anthology/W17-5523

[60] C. Burgers, M. v. Mulken, and P. J. Schellens, "Verbal Irony: Differences in Usage Across Written Genres," *Journal of Language and Social Psychology*, vol. 31, no. 3, pp. 290–310, 2012, _eprint: https://doi.org/10.1177/0261927X12446596. [Online]. Available: https://doi.org/10.1177/0261927X12446596

[61] A. Agrawal, A. An, and M. Papagelis, "Leveraging transitions of emotions for sarcasm detection," in *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2020, pp. 1505–1508.

[62] E. Riloff, A. Qadir, P. Surve, L. De Silva, N. Gilbert, and R. Huang, "Sarcasm as Contrast between a Positive Sentiment and Negative Situation." in *EMNLP*, vol. 13, 2013, pp. 704–714. [Online]. Available: http://www.anthology.aclweb.org/D/D13/D13-1066.pdf

[63] D. Al-Ghadhban, E. Alnkhilan, L. Tatwany, and M. Al-Razgan, "Arabic sarcasm detection in twitter," *2017 International Conference on Engineering and MIS (ICEMIS)*, pp. 1–7, 2017.

[64] M. Bouazizi and T. O. Ohtsuki, "A Pattern-Based Approach for Sarcasm Detection on Twitter," *IEEE Access*, vol. 4, pp. 5477–5488, 2016.

[65] M. Y. Manohar and P. Kulkarni, "Improvement sarcasm analysis using nlp and corpus based approach," in *2017 International Conference on Intelligent Computing and Control Systems (ICICCS)*, 2017, pp. 618–622.

[66] A. Agrawal and A. An, "Unsupervised Emotion Detection from Text Using Semantic and Syntactic Relations," in *2012 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology*, vol. 1, Dec. 2012, pp. 346–353.

[67] C. Goddard, "Interjections and Emotion (with Special Reference to "Surprise" and "Disgust")," *Emotion Review*, vol. 6, no. 1, pp. 53–63, Jan. 2014. [Online]. Available: http://journals.sagepub.com/doi/10.1177/1754073913491843

[68] D. Ghazi, D. Inkpen, and S. Szpakowicz, "Prior and contextual emotion of words in sentential context," *Computer Speech & Language*, vol. 28, no. 1, pp. 76–92, Jan. 2014. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S0885230813000363

[69] H. Xu, M. Lan, and Y. Wu, "ECNU at SemEval-2018 task 1: Emotion intensity prediction using effective features and machine learning models," in *Proceedings of The 12th International Workshop on Semantic Evaluation*. New Orleans, Louisiana: Association for Computational Linguistics, Jun. 2018, pp. 231–235. [Online]. Available: https://www.aclweb.org/anthology/S18-1035

[70] H. Xu, W. Yang, and J. Wang, "Hierarchical emotion classification and emotion component analysis on chinese micro-blog posts," *Expert Syst. Appl.*, vol. 42, pp. 8745–8752, 2015.

[71] F. Benamara, B. Chardon, Y. Mathieu, V. Popescu, and N. Asher, "How do negation and modality impact on opinions?" in *Proceedings of the Workshop on Extra-Propositional Aspects of Meaning in Computational Linguistics*, ser. ExProM '12. Stroudsburg, PA, USA: Association for Computational Linguistics, 2012, pp. 10–18. [Online]. Available: http://dl.acm.org/citation.cfm?id=2392701.2392703

[72] G. A. Miller, "Wordnet: A lexical database for english," *Association for Computing Machinery, Commun. ACM*, vol. 38, no. 11, p. 39–41, Nov. 1995. [Online]. Available: https://doi.org/10.1145/219717.219748

[73] G. Forman, "An extensive empirical study of feature selection metrics for text classification [j]," *Journal of Machine Learning Research - JMLR*, vol. 3, 03 2003.

[74] A. K. Uysal and S. Gunal, "The impact of preprocessing on text classification," *Information Processing and Management*, vol. 50, pp. 104 – 112, 01 2014.

[75] E. Riloff, A. Qadir, P. Surve, L. De Silva, N. Gilbert, and R. Huang, "Sarcasm as Contrast between a Positive Sentiment and Negative Situation." in *EMNLP*, vol. 13, 2013, pp. 704–714. [Online]. Available: http://www.anthology.aclweb.org/D/D13/D13-1066.pdf

[76] C. O. Alm, D. Roth, and R. Sproat, "Emotions from text: Machine learning for text-based emotion prediction," in *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, ser. HLT '05. USA: Association

for Computational Linguistics, 2005, p. 579–586. [Online]. Available: https://doi.org/10.3115/1220575.1220648

[77] L. De Bruyne, O. De Clercq, and V. Hoste, "LT3 at SemEval-2018 task 1: A classifier chain to detect emotions in tweets," in *Proceedings of The 12th International Workshop on Semantic Evaluation*. New Orleans, Louisiana: Association for Computational Linguistics, Jun. 2018, pp. 123–127. [Online]. Available: https://www.aclweb.org/anthology/S18-1016

[78] A. D. S, R. S, S. M. Rajendram, and M. T T, "SSN MLRG1 at SemEval-2018 task 1: Emotion and sentiment intensity detection using rule based feature selection," in *Proceedings of The 12th International Workshop on Semantic Evaluation*. New Orleans, Louisiana: Association for Computational Linguistics, Jun. 2018, pp. 324–328. [Online]. Available: https://www.aclweb.org/anthology/S18-1048

[79] J. H. Park, P. Xu, and P. Fung, "PlusEmo2Vec at SemEval-2018 task 1: Exploiting emotion knowledge from emoji and #hashtags," in *Proceedings of The 12th International Workshop on Semantic Evaluation*. New Orleans, Louisiana: Association for Computational Linguistics, Jun. 2018, pp. 264–272. [Online]. Available: https://www.aclweb.org/anthology/S18-1039

[80] J. Brooke, "A semantic approach to automated text sentiment analysis," Master's thesis, SIMON FRASER UNIVERSITY, British Columbia, BC, Canada, 2009.

[81] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," *Transactions of the Association for Computational Linguistics*, vol. 5, p. 135–146, Dec 2017.

[82] J. Pennington, R. Socher, and C. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 1532–1543.

[83] M. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, 2018.

[84] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts, "Learning word vectors for sentiment analysis," in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Portland, Oregon, USA: Association for Computational Linguistics, June 2011, pp. 142–150. [Online]. Available: http://www.aclweb.org/anthology/P11-1015

[85] P. Nakov, A. Ritter, S. Rosenthal, V. Stoyanov, and F. Sebastiani, "SemEval-2016 task 4: Sentiment analysis in Twitter," in *Proceedings of the 10th International Workshop on Semantic Evaluation*, ser. SemEval '16. San Diego, California: Association for Computational Linguistics, June 2016.

[86] S. Oraby, V. Harrison, L. Reed, E. Hernandez, E. Riloff, and M. Walker, "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*. Los Angeles: Association for Computational Linguistics, Sep. 2016, pp. 31–41. [Online]. Available: https://www.aclweb.org/anthology/W16-3604

[87] R. Misra and P. Arora, "Sarcasm detection using hybrid neural network," *arXiv preprint arXiv:1908.07414*, 2019.

[88] M. Khodak, N. Saunshi, and K. Vodrahalli, "A large self-annotated corpus for sarcasm," *arXiv preprint arXiv:1704.05579*, 2017.

[89] E. Cecilia and A. Ovesdotter, *Affect in text and speech*. ProQuest, Citeseer, 2008.

[90] H. G. Wallbott and K. R. Scherer, "How universal and specific is emotional experience? evidence from 27 countries on five continents," *Information (International Social Science Council)*, vol. 25, no. 4, pp. 763–795, 1986.

[91] H. Schuff, J. Barnes, J. Mohme, S. Padó, and R. Klinger, "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*, 2017, pp. 13–23.

[92] S. M. Mohammad, P. Sobhani, and S. Kiritchenko, "Stance and sentiment in tweets," *ACM Transactions on Internet Technology (TOIT)*, vol. 17, no. 3, p. 26, 2017.

[93] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[94] R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Ng, and C. Potts, "Recursive deep models for semantic compositionality over a sentiment treebank," in *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Seattle, Washington, USA: Association for Computational Linguistics, Oct. 2013, pp. 1631–1642. [Online]. Available: https://aclanthology.org/D13-1170

[95] X. Yang, S. Karaman, J. Tetreault, and A. Jaimes, "Journalistic guidelines aware news image captioning," in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Online and Punta Cana, Dominican Republic: Association for Computational Linguistics, Nov. 2021, pp. 5162–5175. [Online]. Available: https://aclanthology.org/2021.emnlp-main.419

Nastaran Babanejad is a Postdoctoral Research Fellow at the Lassonde School of Engineering, York University, Canada. She received her PhD in Computer Science at Department of Electrical Engineering and Computer Science at York University, Canada. Her research interest includes natural language processing, affect detection, data mining, machine learning, deep neural networks and recommendation systems. Her research has been published in ACL, COLING, ACM RecSys (INRA).

Heidar Davoudi is an Assistant Professor at Ontario Tech University, Canada. Before joining Ontario Tech, he was a postdoctoral research fellow at University of Waterloo. He received his PhD in Computer Science at Department of Electrical Engineering and Computer Science at York University, Canada. His research interest includes data mining, machine learning, and natural language processing. His research has been published in TKDE, Information Retrieval Journal, SIGKDD, SIAM SDM, CIKM, NAACL-HLT, EACL, COLING, and PAKDD.

Ameeta Agrawal is an Assistant Professor in the Department of Computer Science at Portland State University, Portland, USA. Before joining Portland State, she was a postdoctoral researcher in the Data Mining lab at the Department of Electrical Engineering and Computer Science, York University, Canada, where she also obtained her Ph.D. and MSc. in Computer Science. Her research interests are in the areas of natural language processing, machine learning, and computational social sciences.

Aijun An is a Professor in the Department of Electrical Engineering and Computer Science at York University, Toronto, Canada. She received her PhD in Computer Science from the University of Regina and held research positions at the University of Waterloo before joining York University in 2001. Her research interests include data mining, machine learning and natural language processing. She has published widely in premier journals and conferences in these areas. She has been leading various research projects supported by industry and federal and provincial funding agencies in Canada.

Manos Papagelis is an Associate Professor of Electrical Engineering and Computer Science (EECS) at the Lassonde School of Engineering, York University, Canada. His research interests include data mining, graph mining, NLP, machine learning, big data analytics, and knowledge discovery. He holds a Ph.D. in Computer Science from the University of Toronto, Canada, and a M.Sc. and a B.Sc. in Computer Science from the University of Crete, Greece. Prior to joining York University, he was a postdoctoral research fellow at the University of California, Berkeley, research intern at Yahoo! Labs, Barcelona and research fellow at the Institute of Computer Science, FORTH, Greece. He is the recipient of two best paper awards (IEEE MDM 2018; IEEE MDM 2020), an outstanding reviewer award (ACM CIKM 2017) and the EAAI journal 2005-2010 top cited article award. He has lectured at the University of Toronto, the University of California, Berkeley and York University.