Language Independent Sentiment Analysis with Sentiment-Specific Word Embeddings

Carl Saroufim, Akram Almatarky, Mohamed Abdel Hady
Microsoft Corporation, Redmond WA
{casarouf, akrgab, mohabdel}@microsoft.com

Abstract

Data annotation is a critical step to train a text model but it is tedious, expensive and time-consuming. We present a language independent method to train a sentiment polarity model with limited amount of manually-labeled data. Word embeddings such as Word2Vec are efficient at incorporating semantic and syntactic properties of words, yielding good results for document classification. However, these embeddings might map words with opposite polarities, to vectors close to each other. We train Sentiment Specific Word Embeddings (SSWE) on top of an unsupervised Word2Vec model, using either Recurrent Neural Networks (RNN) or Convolutional Neural Networks (CNN) on data auto-labeled as "Positive" or "Negative". For this task, we rely on the universality of emojis and emoticons to auto-label a large number of French tweets using a small set of positive and negative emojis and emoticons. Finally, we apply a transfer learning approach to refine the network weights with a small-size manually-labeled training data set. Experiments are conducted to evaluate the performance of this approach on French sentiment classification using benchmark data sets from SemEval 2016 competition. We were able to achieve a performance improvement by using SSWE over Word2Vec. We also used a graph-based approach for label propagation to auto-generate a sentiment lexicon.

1 Introduction

Text sentiment analysis is defined as the computational study of documents, sentences or phrases (aspect level), to detect opinions, sentiments, emotions, etc. It is particularly useful for companies to collect feedback about their products, analyze the public opinion about their brand, for political parties to monitor the population support, etc. Document-level sentiment analysis corresponds to assigning an overall sentiment polarity to a document. It can be formulated as a two-class classification problem: positive or negative, excluding the neutral case of documents with no polarity (Zhang et al., 2018).

For this task, both supervised and unsupervised learning approaches have been used. Supervised learning methods typically use bag-of-words (which ignores word orders and semantics and suffers from high dimensionality and sparsity), or, more recently, word embeddings, which requires unsupervised training on a big corpus of data. It provides a mapping of words to dense vectors of fixed length, encoding semantic and syntactic properties of those words. The document can then be represented by the average embedding vector of the words it contains. Language-dependent features such as Part of Speech, grammatical analysis, lexicons of opinions and emotions have also been applied successfully (Zhang et al., 2018).

While the use of standard word embedding techniques such as Word2Vec (Mikolov et al., 2013) or C&W (Collobert et al., 2011) can enhance the performance of prediction models and perform well on general classification task, sentiment is not properly encoded in those vectors. According to Tang (2014), words such as "good" and "bad" might be close in the embeddings space, although they have opposite polarities, because of similar usage and grammatical rules. We show in this paper that training Sentiment Specific Word Embeddings (SSWE) by updating an initial Word2Vec model trained on a big corpus of tweets provides better word embeddings for the task of sentiment classification. SSWE training is performed by updating word embeddings as part of a supervised deep learning framework, by training a model on sentiment-labeled data. Through backpropagation, the weights of the Embedding Layer will be adjusted and incorporate sentiment. At the end of the training, the embedding matrix can be extracted from the model and will be used

Proceedings of the 9th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis, pages 14–23
Brussels, Belgium, October 31, 2018.
https://doi.org/10.18653/v1/P17

to featurize words and documents. As a result, this process needs a big amount of data labeled with "Positive" and "Negative". While labeled data sets and sentiment lexicons exist in English and can be used to label a big number of documents to build the SSWE (e.g. SentiWordNet or ANEW in English as lexicons), they are scarce in other languages (Pak and Paroubek, 2011). Hence the need to find a systematic way of labeling a big number of documents without any language knowledge, then let prediction models "learn" from them.

One way to do it, is to rely on a universal opinion lexicon that would hold true in any language. With the rise of social media, the widespread use of emojis[1] provide us with such a tool. In particular, Twitter is one of the biggest online social media where users post over 500 million "tweets" every day, with a frequent use of emojis[2]. A tweet is a short (up to 140 characters) user-generated text, typically noisy and written in a very casual language.

By accessing and querying a big number of tweets of a specific language and which have specific emojis, we can auto-label them based on the polarity of those emojis. The assumption is that if emojis are found in a tweet, then it expresses a sentiment (ie. it is not neutral) which has the same polarity as the emoji. This method is called distant-supervised learning and has been applied successfully in several similar scenarios (Pak and Paroubek, 2011; Tang et al., 2014; Narr et al., 2012).

In this paper, we focus on French, assuming no language knowledge. The SSWE we get through the described methodology is then used to featurize documents when training a prediction model on a new French data set for sentiment analysis. The major contributions of the work presented in this paper are:

- We develop SSWE models by updating Word2Vec embeddings trained on tweets, through supervised learning with Recurrent Neural Networks (RNN) and Convolutional Neural Networks (CNN), on a big data set of tweets auto-labeled through emojis.
- We show that SSWE perform better than the underlying Word2Vec embeddings, even on data sets different and much less noisy than tweets. SSWE trained with auto-labeled tweets from which the emojis were removed, improve the sentiment prediction on data sets that have no or little emojis.
- We show that transfer learning starting with the deep learning model used to train the SSWE performs better than training a new traditional ML prediction model from scratch and using SSWE as features.

In Section 2, we present a literature review. In Section 3, we present the methodology used to auto-label tweets, train a Word2Vec model and update it into SSWE. Section 4 describes the experiment we conducted and discusses results. Finally, in section 5 we summarize our results and present directions for future work.

2 Related Work

We present here a brief review of previous work for sentiment classification. Most early research follows Pang (2002) who uses bag of words and one-hot-encoding to represent words in movie reviews, using linguistic features and various classifiers. Work by Pang (2008), and Owoputi (2013) focused on the features for learning and their effectiveness, such as part-of-speech, syntax, negation handling and topic-oriented features. Mohammad (2013), achieved the best results for the sentiment classification task of the SemEval competition by using sentiment lexicons and handcrafted rules. Gamon (2004) investigated the sentiment analysis task on noisy data collected from customer feedback surveys, using lexical (lemmatized n-gram) and linguistic features (part-of-speech tagging, context free phrase structure patterns, transitivity of a predicate, tense information etc.). They also tried feature reduction and applied SVM model for classification.

Another approach for features representation is to use low-dimensional real-valued vectors (word embeddings) to represent the words, such as Word2Vec or C&W. Maas (2011) proposed the use of probabilistic document models with a sentiment predictor function for each word. Bespalov (2012) relied on latent semantic analysis to initialize embeddings. Labutov (2013) relies on pre-trained word embeddings which are re-embedded in a supervised task using labeled data by performing unconstrained optimization of a convex objective.

[1] In this paper, we will use the word "emojis" to refer to both "emojis" and "emoticons".

[2] Twitter statistics: http://www.internetlivestats.com/twitter-statistics

Tang (2014) proposed to update word embeddings specifically for sentiment classification, arguing that words with opposite polarities might end up being neighbors. They learn word embeddings from a massive number of tweets collected by a distant-supervised way based on the existence of positive and negative emojis. They propose a sentiment-specific word embedding (SSWE) model that is a modification of C&W to predict not just the lexical form (n-gram) but also the sentiment of the word in that context. They were able to produce comparable results to the top system (rule-based) in the SemEval 2013 competition.

Ren (2016) extended the work of Tang (2014), arguing that the topic information of a tweet affects the sentiment of its words. They modified the learning objective of the C&W model to also incorporate the sentiment information as well as the topic distribution provided by LDA models. They also tackle the problem of word polysemy (words that have multiple meanings based on their context) by creating context representation for each word occurrence (environment vector) and cluster these vectors into ten groups using k-means algorithm. The words get their meaning by their distance to the centroids of the clusters. Finally, they train a CNN for sentiment classification.

In terms of classification models, with the regained interest in deep learning, recent work shifted to the use of RNNs. These networks process every element of a sequence in a way depending of all previous computations, keeping track of previous information across the sequence. In particular, LSTM (Long Short-Term Memory) can learn long-term dependencies and is thus popular for sentiment classification of sentences. Gugilla (2016) uses both LSTM and CNN with Word2Vec and linguistic embeddings to distinguish between neutral and sentiment documents. Qian (2017) uses LSTM with linguistic resources such as sentiment lexicon, negation and intensity words, to help identify the sentiment effect in sentences. Xu (2016) proposed the use of cached LSTM model to further enhance the capabilities of LSTM to capture the global semantic features and the local semantic features for long text sentiment classification.

For Twitter sentiment classification using distant supervision, while some research used lexicon-based approaches with positive and negative sentiment words (Taboada et al., 2011; Thelwall et al., 2012), other studies leveraged emojis for distant supervision (Pak and Paroubek, 2011; Zhao et al., 2012). To ensure repeatability of experiments in many languages without any language knowledge, we will also take this direction even though it means we are neglecting precious language-dependent features that would have increased the prediction power of our models.

	# Raw Tweets (in M)	**# Tweets cleaned (in M)**	**% Positive in cleaned**
April 18	4,14	2,62	51.29
May 18	3,89	2,44	49.20
June 18	2,83	1,76	50.94
Total	10,86	6,82	50.43

Table 1: Characteristics of the auto-labeled tweets used in this paper.

3 SSWE Training

3.1 Auto-Labeled Tweets and Preprocessing

Having access to a big database of tweets, we first queried three months of French tweets: April, May and June 2018 (partial). Only tweets containing one or more of a specified list of emojis are considered, but all emojis in a tweet should have the same polarity (confusing tweets are discarded). This polarity will correspond to the label of the tweet. For example, :) is a positive emoji, while :(is negative. To avoid dealing with class imbalance, we downsample the majority class to get a 1:1 ratio of positive to negative auto-labeled tweets.

We perform the following operations on the auto-labeled tweets (**Preprocessing 0**). Using regular expression (Regex), we remove "RT" (corresponding to Retweets), "@" and name mentions, tweet links and duplicate tweets. We also separate emojis when there is no space between them. Otherwise, multiple stacked emojis would be seen as one word instead of a succession of emojis by standard open-source tokenizers. Finally, we replace emojis involving punctuation by "Emoji_01", "Emoji_02", etc. Otherwise, standard tokenizers would separate punctuation marks and break emojis. For example: ":P" would be tokenized into [":", "P"].

Figures 1 and 2 respectively show the positive and negative emojis used for auto-labeling.

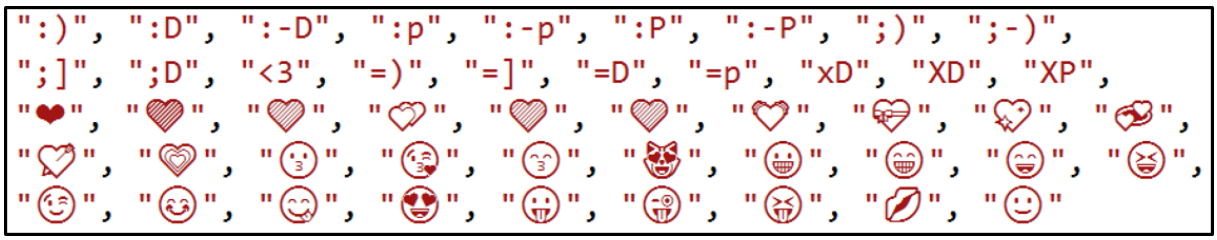

Figure 1: List of positive emojis used for auto-labeling.

Figure 2: List of negative emojis used for auto-labeling.

Table 1 summarizes statistics about the cleaned data sets.

We explored two additional preprocessing methods to study their impact on sentiment prediction.

- **Preprocessing A**: Remove emojis after auto-annotation to avoid biasing the prediction models into classifying mostly based on emojis.
- **Preprocessing B**: Replace every occurrence of three or more successive characters into two of them.

The reasoning behind **Preprocessing B** is as follows: since tweets contain casual language, users sometimes repeat the same character many times to stress out an emotion. For example, the word "aime" could appear as "aime", "aiime", "aiiime", "aiiiiiime", etc. As a result, the number of features will increase when using ngrams, and the dictionary size of word embeddings would also increase. This would dilute the word embeddings because the same word would be considered as different ones which will carry a lower weight than the case where all words with repetitions are assumed to be one. Assuming that when the letter is repeated 2 or more times, the number of repetitions is random and expresses the same meaning, we experiment with replacing any occurrence of two or more successive characters with two occurrences.

3.2 Word2Vec Model Training

Twitter sentiment classification has traditionally been conducted through machine learning models using labor intensive features. A less labor-intensive feature engineering approach has been to rely on word embeddings such as Word2Vec, which incorporates syntactic context of words. After preprocessing the autolabeled tweets, we trained two Word2Vec models on them in an unsupervised way. One where only standard preprocessing is applied to autolabeled data (**Preprocessing 0**), and the second on autolabeled data where characters repetitions above three are removed (**Preprocessing B**). Note that since the autolabeled data is divided in three files, and to avoid loading all the tweets in-memory at once, Word2Vec is trained incrementally on the three files.

3.3 Sentiment Specific Word Embeddings Training

While Word2Vec typically performs well on general classification tasks, it might not be effective enough for sentiment classification because it ignores word sentiments. Hence the need to update the trained Word2Vec embeddings to incorporate sentiment. We train a deep learning network (either RNN or CNN) using first an Embedding layer, initialized with the trained Word2Vec embeddings. During training on the autolabeled tweets, network weights, including word embeddings, are updated. At the end of training, we extract the embedding matrix: it has the same vocabulary as the original Word2Vec, but weights now incorporate sentiment information. We call this embedding matrix a "Sentiment Specific Word Embeddings" (SSWE) (Tang et al., 2014). The assumption to test is if indeed, SSWE used as features would perform better than Word2Vec for the task of sentiment analysis.

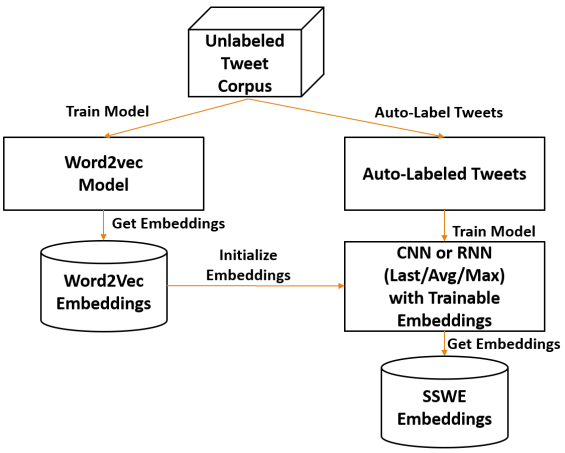

Figure 3: Methodology used to train the SSWE.

We trained four different types of models:

– SSWE_C trained using a CNN structure (three parallel convolutional layers and global max pooling, then dense layer with ReLu, then dense layer with softmax).

– SSWE_R, SSWE_R_avg, and SSWE_R_max trained using an RNN structure (two bidirectional LSTM from which respectively the last output, average of outputs or max of outputs goes through a dense layer with softmax).

After performing **Preprocessing A**, we can train four additional models by removing emojis from the autolabeled data when updating the embeddings of Word2Vec. Note that in this case, emojis still have embeddings but they are not updated during the training of SSWE in order to avoid biasing the weights into relying mostly on emojis.

After performing **Preprocessing B**, we can get an additional eight models by using the apropriate trained Word2Vec (without character repetitions).

Note that since the auto-labeled data is divided in three files, and to avoid loading all the tweets in-memory at once, SSWE is trained incrementally on the three files. For each one, we used three epochs and a batch size of 500.

Figure 3 summarizes the methodology used to train the SSWE.

Figure 4[3] and Figure 5 summarize the architectures of the deep learning models used.

[3] http://www.wildml.com/2015/11/understanding-convolutional-neural-networks-for-nlp/

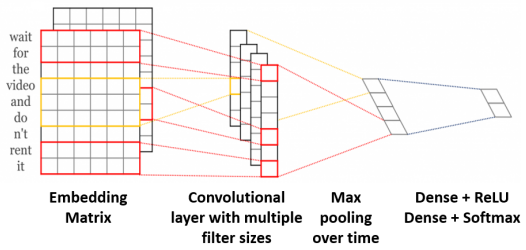

Figure 4: CNN architecture.

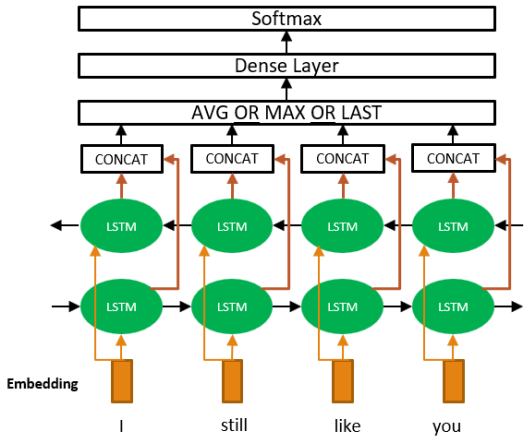

Figure 5: RNN architecture.

3.4 Graph-Based Label Propagation

We used the method introduced by Hamilton (2016) to create a dictionary of positive and negative words from Word2Vec embeddings in an unsupervised way. The idea revolves around building word embeddings (the paper uses Vector Space Model instead of Word2Vec), then creating a graph of words connected to their k-nearest neighbors in the embeddings space with edges weighted by the cosine similarity. Using a short list of "seed words" for positive and negative polarities, sentiment is propagated through the network through a random walk method. This idea was implemented by Hamilton et al. in a Python package called SENTPROP.

We use emojis as seed words and Word2Vec trained on tweets from which we only kept a specific short set of positive and negative emojis (so the focus is on actual words that are neighbors to the seed emojis instead of other emojis).

4 Experiments and Results

4.1 Setup

We conduct experiments to evaluate the validity of using SSWE over Word2Vec for French sentiment classification. In addition to the auto-labeled tweets, four manually labeled French data sets were used. The first three data sets come from the SemEval 2016 competition (International Workshop on Semantic Evaluation). It is an ongoing series of evaluations of computational semantic analysis systems, organized under the umbrella of SIGLEX, the Special Interest Group on the Lexicon of the Association for Computational Linguistics. The first data set (Train) corresponds to French Restaurant Reviews. The two others correspond to French Restaurant Reviews (Test1; i.e. Same domain as the training set) and French Museum Reviews (Test2). The latter being out-of-domain, it is a good gauge of how generalizable a model trained on the training set is to different French data sets. Those three data sets have been labeled for Aspect Based Sentiment Analysis. It is the task of mining and summarizing opinions from text about specific entities and their aspects (Apidianaki et al., 2016) they are annotated with relevant entities, aspects and polarity values. In order to use those three data sets in the context of document level sentiment classification, we ungrouped documents into sentences, then filtered them:

- We reject the sentences with mixed polarity values.

- We keep the rest of the sentences and label them as "Positive" if all polarity values are positive and as "Negative" if all polarity values are negative.

Note that we were not able to download all the reviews as some of them were not available anymore[4]. The fourth data set is comprised of manually labeled French tweets that are not publicly available (Test3). These tweets do not contain emojis, and thus the performance on this data set will solely rely on the understanding that our trained models will have of the French language. The four French data sets (after processing) and their characteristics are summarized in Table 2. We can see that they are all relatively well-balanced, so no special treatment for class imbalance will be performed.

[4]The data can be downloaded from http://alt.qcri.org/semeval2016/task5/index.php?id=data-and-tools

Data Set	Naming	# Doc	% Positive
SemEval 2016 Restaurant Train	Train	1338	47.80
SemEval 2016 Restaurant Test	Test1	515	47.60
SemEval 2016 Museum Test	Test2	529	54.25
Tweets Manually Labeled	Test3	3296	48.90

Table 2: Characteristics of the training and testing sets used.

4.2 Experiments

The experiments conducted and their results are summarized in Table 3.

In order to establish the importance of the SSWE for sentiment analysis, we trained multiple models using standard features (word ngrams, character ngrams) or more advanced features (Word2Vec, SSWE: average sentence embedding) with standard classifiers (logistic regression (LR), SVM, Random Forest) or deep learning frameworks (CNN or RNN with transfer learning on the data set Train), then evaluated them on Test1, Test2, and Test3. More precisely, we used the following approaches as baselines, training a Logistic Regression (LR), SVM and Random Forest with word and character ngrams, Word2Vec, or both (experiments 1-3).

The three baselines were compared to four methods using SSWE trained with either RNN (taking the last output of the recurrent sequence) or CNN trained on the auto-labeled data and used for feature extraction on Train. We compared the performance with SSWE trained on auto-labeled data cleaned from the emojis to prevent the model from giving a high weight to emojis (experiments 4-7).

We compared those approaches with the use of average or maximum of the output sequences of the RNN (trained on auto-labeled data without emoji, then used for feature engineering on Train) (experiments 8-9).

We then add the character and ngrams features (experiments 10-15).

Finally, we compared them with deep learning and transfer learning. Keeping the same network that was used to train the SSWE, we fix the SSWE and let the last layer(s) be trainable on Train. For the RNN, the dense layer and softmax are trainable. For the CNN, training only the dense layer and softmax as part of transfer learning yields poor

results, so we train the dense layer and ReLu as well as the last dense layer and softmax during transfer learning. The reason why transfer learning is used instead of direct training on Train is the lack of data: Train having only 1338 unique documents, this is not enough to tune a deep learning model which has a much larger number of parameters. (experiments 16-21)

We also explored two additional methods that do not require any knowledge of the language and that will be detailed in subsequent sections:

1) using a dictionary auto-built by label propagation using specific emoji seeds on the Word2Vec model (experiments 22-24).

2) repeating the experiment on data for which two or more repetitions of characters are replaced with two characters.

All approaches with Logistic Regression, SVM or Random Forest used the Scikit-Learn Python package. Word2Vec models were built using genism, while SSWE and deep learning frameworks used Keras with Tensorflow backend.

4.3 Evaluation Methods

For models trained with Logistic Regression, SVM and Random Forest, we decided to report and compare only results obtained with Logistic Regression for fair comparison. The reported results correspond to those achieved with a set of parameters giving the best results on part of the training set, through sweeping. The parameters that were swept on in Logistic Regression are: size of word ngrams, size of character ngrams, lowercasing of words, maximum number of iterations for the 'lbfgs' solver, inverse of regularization strength for 'l2' penalty. For models trained with Deep Learning, multiple combinations of epochs and batch sizes were used, and we report here the best results obtained.

While F-score and Accuracy (since there is no class imbalance) can be fair ways of evaluating the performance, we decided to rely mainly on the AUC (Area under the Curve) since it is independent of the classification threshold choice and is a good indicator of the ability of the models to distinguish between Positive and Negative examples.

4.4 Results and Analysis

With Logistic Regression, using only ngram features (1) give the worse results on Test2 and Test3. This is probably because ngrams obtained on a specific training set fail to generalize well to out-of-domain data sets (Test2 and Test3). Using Word2Vec only (2) is slightly worse on Test1 but generalizes better on out-of-domain data sets. Using both Word2Vec and ngram (3) considerably improves the results compared to (1) and (2), showing the importance of Word2Vec in adding context information in word features, which ngram does not. This will be the main baseline to compare to the use of SSWE.

	Features	Test1	Test2	Test3
LOGISTIC REGRESSION				
1	ngram	86.27	75.36	72.49
2	Word2Vec	86.03	79.00	86.75
3	ngram + Word2Vec	90.18	83.3	86.71
4	SSWE_R_last	86.11	79.25	84.28
5	SSWE_R_last_no_emoji	87.86	83.12	86.52
6	SSWE_C	85.34	78.91	83.72
7	SSWE_C_no_emoji	89.15	83.59	87.83
8	SSWE_R_avg_no_emoji	88.01	83.07	87.17
9	SSWE_R_max_no_emoji	86.29	82.49	85.40
10	ngram + SSWE_R_last	89.86	83.10	85.06
11	ngram + SSWE_R_last_no_emoji	90.79	84.72	85.82
12	ngram + SSWE_C	89.05	83.62	84.24
13	ngram + SSWE_C_no_emoji	**91.42**	**85.67**	**87.50**
14	ngram + SSWE_R_avg_no_emoji	90.84	84.53	86.90
15	ngram + SSWE_R_max_no_emoji	91.20	85.75	87.37
TRANSFER LEARNING				
16	SSWE_R_last	90.17	84.2	83.89
17	SSWE_R_last_no_emoji	**92.78**	**91.02**	**90.59**
18	SSWE_C	81.31	76.87	79.91
19	SSWE_C_no_emoji	89.97	87.65	89.27
20	SSWE_R_avg_no_emoji	91.86	90.71	90.03
21	SSWE_R_max_no_emoji	91.61	91.13	90.34
WITH DICTIONARY (Dict)				
22	Dict	63.52	63.86	64.85
23	LR + ngram + Dict	88.95	76.97	77.73
24	LR + ngram + Dict + SSWE_C_no_emoji	91.21	85.73	87.38

Table 3: Experiments Results with Logistic Regression, Transfer Learning and Dictionary Lookup in terms of AUC (in %).

We notice that SSWE trained on the auto-labeled data where emojis were kept (4), (6), yields results similar or slightly lower than using Word2Vec only (2). The same comparison holds when we add ngrams: (10) and (12) vs. (3). This is probably because the labels of the auto-labeled data are directly correlated to specific positive and negative emojis since they were used for autolabeling. As a result, training SSWE on the auto-labeled data without removing emojis likely biased the embeddings into giving higher weights

to emojis. Since all the test sets have little to no emojis, the use of these SSWE yields poor results.

SSWE trained on auto-labeled data cleaned from the emojis (without ngrams) in (5), (7), (8) and (9) provides similar or better performance than Word2Vec only (2). Adding ngrams to these SSWE in (11), (13), (14) and (15) gives the best results with Logistic Regression compared to the best baseline (3). While beating the performance of (3) on Test3 seems hard (likely due to the fact that Word2Vec was trained on data similar to Test3 since they are both tweets), ngrams and SSWE trained as part of a CNN model on auto-labeled data without emojis (13) gives on average the best results on the three data sets when using Logistic Regression, compared to (3): 1.34%, 2.37%, and 0.79% improvement on Test1, Test2, Test3 respectively. These results already confirm that SSWE improves the prediction performance in sentiment analysis over Word2Vec, likely because it incorporates sentiment polarity.

These results were even more confirmed with deep learning. SSWE trained on auto-labeled data with emojis also perform less good than their counterpart trained without emojis: (16) vs. (17) and (18) vs. (19). The best results are obtained on average across the three test sets when transfer learning is applied with the RNN, using the last output from the bidirectional LSTM. Compared to the best baseline (3), we achieve: 2.6%, 7.72%, and 3.88% improvement on Test1, Test2, Test3 respectively.

We also used the scores of the auto-generated dictionary with a 0.5 threshold to assign a polarity to the words. We then classify the documents by comparing the distribution of negative and positive words. This gave worse results than ngram with logistic regression ((1) vs. (22)). We also used the word scores to create a dictionary with 10 labels: if the word has a score between 0 and 0.1, it is assigned to the sentiment_bin_1 class; between 0.1 and 0.2, it is assigned to sentiment_bin_2, etc. The distribution among those ten classes is then used as an additional feature in our experiments with logistic regression. However, this feature does not improve the results((1) vs. (23) and (13) vs. (24)).

Finally, we did not notice an improvement when removing repetition of characters above three. This might be explained by the fact that character repetition can be important for emphasis: the same word with more character repetitions might have a higher polarity. For example, "heureuxxxxx" might show more excitement and be "more positive" than "heureuxx".

4.5 Discussion

Given those results, multiple conclusions can be drawn:

- Word2Vec boosts performance of ngrams, especially on out-of-domain testing set.
- SSWE trained on data autolabeled with emojis and where emojis were not removed, negatively impacts the performance of the model on data that have little or no emojis.
- SSWE trained on data autolabeled with emojis and where emojis were removed, provides an improvement over Word2Vec.
- Transfer Learning in deep learning by fixing the network structure (including SSWE trained without emojis) on the training set, yields the best improvements over the baseline with an increase of more than 7% AUC on Test2 for example. This is the most noticeable when using RNN and taking the last output of the bidirectional LSTM sequence.

5 Conclusion and Future Work

In this paper, we propose to label a big number of tweets in any language (here French) using a small set of positive and negative emojis, train a Word2Vec model on the tweets, then update the embeddings through deep learning with bidirectional LSTM on the autolabeled data. The embeddings we get are then enriched with sentiment information and can be used as features for new data sets in the same language. If those data sets have little or no emoji, the embeddings enrichment should be performed using autolabeled data filtered from emojis in order to avoid biasing the embeddings and relying mostly on emojis. We show that these sentiment specific word embeddings perform better than plain Word2Vec using SemEval 2016 French data of restaurant reviews (Train) to train a model, another SemEval 2016 testing set of restaurant reviews (Test1), another of museum reviews (Test2) and a manually labeled data set of French tweets (Test3). It provides an AUC improvement of 1.34%, 2.37%, and 0.79% on Test1, Test2, Test3 respectively with Logistic Regression. The improvement gets to 2.6%,

7.72%, and 3.88% with transfer learning using RNN with the network used to train the SSWE. This methodology can be applied to any language since it does not rely on linguistic features.

This work has a few limitations, which could be better addressed in future work. This includes the use of sentiment lexicons, stemming, normalization, and negation handling. Our work did not explore the graph propagation technique with different embeddings models such as Vector Space Models. In addition, distant-supervised learning was applied on tweets to get labeled data and then train the SSWE. Tweets being written in a very casual language, the results on SemEval data (which is clean) might improve if the SSWE were trained on a more general data set. An area for future work would be to explore distant supervised learning on movie or product reviews, then compare the results with the created SSWE on SemEval data with those obtained using the Twitter SSWE. Finally, our work excludes the case of neutral text. We can specify two thresholds for the polarity prediction and assign a document to the neutral class when the model yields a predicted probability between those two thresholds. Another method is to train a subjectivity classifier followed by a polarity classifier for subjective documents.

References

Marianna Apidianaki, Xavier Tannier, and Cécile Richart. 2016. Datasets for aspect-based sentiment analysis in french. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, Paris, France. European Language Resources Association (ELRA).

Dmitriy Bespalov, Yanjun Qi, Bing Bai, and Ali Shokoufandeh. 2012. Sentiment classification with supervised sequence embedding. In *Proceedings of the 2012 European Conference on Machine Learning and Knowledge Discovery in Databases - Volume Part I*, ECML PKDD'12, pages 159–174, Berlin, Heidelberg. Springer-Verlag.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel P. Kuksa. 2011. Natural language processing (almost) from scratch. *CoRR*, abs/1103.0398.

Michael Gamon. 2004. Sentiment classification on customer feedback data: Noisy data, large feature vectors, and the role of linguistic analysis. In *Proceedings of the 20th International Conference on Computational Linguistics*, COLING '04, Stroudsburg, PA, USA. Association for Computational Linguistics.

Chinnappa Guggilla, Tristan Miller, and Iryna Gurevych. 2016. Cnn- and lstm-based claim classification in online user comments. In *COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2740–2751.

William L. Hamilton, Kevin Clark, Jure Leskovec, and Dan Jurafsky. 2016. Inducing domain-specific sentiment lexicons from unlabeled corpora. *CoRR*, abs/1606.02820.

Igor Labutov and Hod Lipson. 2013. Re-embedding words. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 489–493.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 142–150, Stroudsburg, PA, USA. Association for Computational Linguistics.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. *CoRR*, abs/1310.4546.

Saif M. Mohammad, Svetlana Kiritchenko, and Xiaodan Zhu. 2013. Nrc-canada: Building the state-of-the-art in sentiment analysis of tweets. *CoRR*, abs/1308.6242.

Sascha Narr, Michael Hulfenhaus, and Sahin Albayrak. 2012. Language-independent twitter sentiment analysis. *Knowledge discovery and machine learning (KDML), LWA*, pages 12–14.

Olutobi Owoputi, Brendan O'Connor, Chris Dyer, Kevin Gimpel, Nathan Schneider, and Noah A. Smith. 2013. Improved part-of-speech tagging for online conversational text with word clusters. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 380–390. Association for Computational Linguistics.

Alexander Pak and Patrick Paroubek. 2011. Twitter for sentiment analysis: When language resources are not available. In *Proceedings of the 2011 22Nd International Workshop on Database and Expert Systems Applications*, DEXA '11, pages 111–115, Washington, DC, USA. IEEE Computer Society.

Bo Pang and Lillian Lee. 2008. Opinion mining and sentiment analysis. *Found. Trends Inf. Retr.*, 2(1-2):1–135.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up? sentiment classification using machine learning techniques. In *Proceedings of EMNLP*, pages 79–86.

Qiao Qian, Minlie Huang, Jinhao Lei, and Xiaoyan Zhu. 2017. Linguistically regularized lstm for sentiment classification. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL 2017*.

Yafeng Ren, Yue Zhang, Meishan Zhang, and Donghong Ji. 2016. Improving twitter sentiment classification using topic-enriched multi-prototype word embeddings.

Maite Taboada, Julian Brooke, Milan Tofiloski, Kimberly D. Voll, and Manfred Stede. 2011. Lexicon-based methods for sentiment analysis. 37:267–307.

Duyu Tang, Furu Wei, Nan Yang, Ming Zhou, Ting Liu, and Bing Qin. 2014. Learning sentiment-specific word embedding for twitter sentiment classification. In *52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014 - Proceedings of the Conference*, volume 1, pages 1555–1565.

Mike Thelwall, Kevan Buckley, and Georgios Paltoglou. 2012. Sentiment strength detection for the social web. *J. Am. Soc. Inf. Sci. Technol.*, 63(1):163–173.

Jiacheng Xu, Danlu Chen, Xipeng Qiu, and Xuanjing Huang. 2016. Cached long short-term memory neural networks for document-level sentiment classification. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1660–1669. Association for Computational Linguistics.

Lei Zhang, Shuai Wang, and Bing Liu. 2018. Deep learning for sentiment analysis : A survey. *CoRR*, abs/1801.07883.

Jichang Zhao, Li Dong, Junjie Wu, and Ke Xu. 2012. Moodlens: An emoticon-based sentiment analysis system for chinese tweets. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '12, pages 1528–1531, New York, NY, USA. ACM.