Leveraging Large Amounts of Weakly Supervised Data for Multi-Language Sentiment Classification

Jan Deriu
ZHAW

Aurelien Lucchi
ETH Zurich

Valeria De Luca
ETH Zurich

Aliaksei Severyn
Google Research

Simon Müller
SpinningBytes AG

Mark Cieliebak
SpinningBytes AG

Thomas Hofmann
ETH Zurich

Martin Jaggi
EPFL

ABSTRACT

This paper presents a novel approach for multi-lingual sentiment classification in short texts. This is a challenging task as the amount of training data in languages other than English is very limited. Previously proposed multi-lingual approaches typically require to establish a correspondence to English for which powerful classifiers are already available. In contrast, our method does not require such supervision. We leverage large amounts of weakly-supervised data in various languages to train a multi-layer convolutional network and demonstrate the importance of using pre-training of such networks. We thoroughly evaluate our approach on various multi-lingual datasets, including the recent SemEval-2016 sentiment prediction benchmark (Task 4), where we achieved state-of-the-art performance. We also compare the performance of our model trained individually for each language to a variant trained for all languages at once. We show that the latter model reaches slightly worse - but still acceptable - performance when compared to the single language model, while benefiting from better generalization properties across languages.

Keywords

Sentiment classification; multi-language; weak supervision; neural networks

1. INTRODUCTION

Automatic sentiment analysis is a fundamental problem in natural language processing (NLP). A huge volume of opinionated text is currently available on social media. On Twitter alone, 500 million tweets are published every day. Being able to manually process such a high volume of data is beyond our abilities, thus clearly highlighting the need for automatically understanding the polarity and meaning of these texts. Although there have been several progresses towards this goal, automatic sentiment analysis is still a challenging task due to the complexity of human language, where the use of rhetorical constructions such as sarcasm and irony easily confuse sentiment classifiers. Contextualization and informal language, which are often adopted on social media, are additional complicating factors. The Internet is also multi-lingual and each language has its own grammar and syntactic rules.

Given all these difficulties, it is not surprising that the performance of existing commercial systems is still rather poor, as shown in several recent studies [6, 27]. The benchmark work of Ribeiro *et al.* [27] showed that even the performance of the best systems largely varies across datasets and overall leaves much room for improvement. Hence it is important to design a method that generalizes well to different domains and languages.

Contributions. The majority of current research efforts in sentiment analysis focuses on the English language. This is partially due to the large number of resources available in English, including sentiment dictionaries, annotated corpora and even benchmark datasets. An example is the SemEval competition, which is one of the largest competitions on semantic text evaluation and covers several tasks for sentiment analysis [24].

However, only 26.3% of the total number of internet users in 2016 are English speakers [15] and only 34% of all tweets are written in English [21]. Hence there is a strong incentive to develop methods that work well with other languages. In this work, we focus on the question of how sentiment analysis can be done for multiple languages by leveraging existing technologies. Our method is the state-of-the-art approach for sentiment analysis on Twitter data which recently won the SemEval-2016 competition [9]. Here we additionally explore how to best adapt this approach to other languages. The core component of our system is a multi-layer convolutional neural network (CNN), trained in three phases: i) unsupervised phase, where word embeddings are created on a large corpus of unlabeled tweets; ii) distant supervised phase, where the network is trained on a weakly-labeled dataset of tweets containing emoticons; and iii) supervised phase, where the network is finally trained on manually annotated tweets. For English, this system achieved an F1-score of 62.7% on the test data of SemEval-2016 [9].

Although existing CNN approaches [31, 9] can a-priori be trained on any language other than English, these nevertheless require a large amount of training data. Yet resources in languages other than English are lacking, and manually labeling tweets is a time-consuming and expensive process. Two straightforward solutions that do not require manual work can be envisioned: (1) automatically translate the data into English and run the existing English classifier; or (2) train a CNN using only weakly-labeled tweets without using any supervised data. It is expected that a fully-trained CNN would perform better than the aforementioned cases. How-

WWW 2017, April 3–7, 2017, Perth, Australia.
ACM 978-1-4503-4913-0/17/04.
http://dx.doi.org/10.1145/3038912.3052611

ever, it is unclear if such improvement is significant and justifies the need of manually labeling thousands of tweets.

In this paper, we investigate how to effectively train and optimize a CNN for multi-lingual sentiment analysis. We compare the performance of various approaches for non-English texts. In details, our main contributions are:

- An evaluation of the state-of-the-art CNN approach similar to the one proposed in [9] on three new languages, namely French, German and Italian
- A thorough analysis of the influence of network parameters (number of layers, hyper-parameters) and other factors, e.g. the amount of distant-supervised and supervised data, on end-to-end performance
- For each language, a comparison of various approaches for sentiment analysis: (i) full training of the CNN for the considered language; and (ii) automatically translating the texts into a language (English) where a sentiment classifier already exists. Other baseline methods, described in the experimental section, are also compared
- In addition, we show that a single CNN model can be successfully trained for the joined task on all languages, as opposed to separate networks for each individual language. This approach has the advantages of removing the reliance on (possibly inaccurate) language identification systems and it can be easily extended to new languages and multi-language texts. We provide detailed comparison to similar per-language models, and show that the proposed joint model still performs relatively well
- Public release of the source code as well as pre-trained models for all languages tested in this paper, on `http://github.com/spinningbytes/deep-mlsa`

2. RELATED WORK

In the following, we provide an overview of the most relevant works, related to the application of neural networks to sentiment classification, distant supervision and training multi-lingual text classifiers.

Neural networks. Neural networks have shown great promise in NLP over the past few years. Examples are in semantic analysis [33], machine translation [12] and sentiment analysis [34]. In particular, shallow CNNs have recently improved the state-of-the-art in text polarity classification demonstrating a significant increase in terms of accuracy compared to previous state-of-the-art techniques [18, 17, 10, 32, 16, 28, 9]. These successful CNN models are characterized by a set of convolution filters acting as a sliding window over the input word sequence, typically followed by a pooling operation (such as max-pooling) to generate a fixed-vector representation of the input sentence.

CNNs vs RNNs. Recently, recurrent neural network architectures (RNNs), such as long short-term memory networks (LSTMs), have received significant attention for various NLP tasks. Yet these have so far not outperformed convolutional architectures on polarity prediction [29, Table 4]. This has been evidenced by the recent SemEval-2016 challenge [24], where systems relying on convolutional networks rank at the top. In fact, long-term relationships captured well by LSTMs are of minor importance to the sentiment analysis of short tweets. On the contrary, learning powerful n-gram feature extractors (which convolutional networks handle very well) contributes much more to the discriminative power of the model, since these are able to effectively detect sentiment cues. Additionally, LSTMs are much more computationally expensive than CNNs, preventing their application to very large collections like the one used in this paper (hundreds of millions tweets).

Distant-supervised learning. The use of semi-supervised or unsupervised learning has been an active research direction in machine learning and particularly for various NLP applications. There is empirical evidence that unsupervised training can be beneficial for supervised machine learning tasks [11]. In this paper, we consider a variant of unsupervised learning named distant pre-training which consists in inferring weak labels from data without manual labels. This approach has been used for text polarity classification where significantly larger training sets were generated from texts containing emoticons [13, 32]. Severyn and Moschitti [32] have shown that training a CNN on these larger datasets, followed by additional supervised training on a smaller set of manually annotated labels, yields improved performance on tweets.

Multi-language sentiment classification. Sentiment classification has drawn a lot of attention in the past few years both in industry and academia [24, 6]. Yet most of the research effort has been focusing on tweets written in one language (mostly English). One exception is the work of Boiy and Moens [4] that studied the portability of a learned sentiment classification model across domains and languages. They focused on French, Dutch and English, and showed that significant disparities between these languages can severely hinder the performance of a classifier trained on hand-crafted features.

The major factor that limits the development of accurate models for multi-lingual sentiment analysis is the lack of supervised corpora [2, 7]. Most of the existing approaches addressing this problem [22, 1] try to transfer knowledge from English – for which tools, labelled data and resources are abundant – to other languages for which resources are rather limited. An example is the approach introduced in [22], which transfers hand-crafted subjectivity annotation resources - such as a per-word sentiment lexicon - from English to Romanian. A similar approach introduced in [1] consists in translating a target language to English and then to use an English sentiment classifier rather than one specific to the target language. Several approaches have also been proposed to build distributed representations of words in multiple languages. The work of Wick *et al.* [35] used a Wikipedia corpus of five languages to train word embeddings, and then used anchor terms (names, cross-lingual words) to align the embeddings. Gouws *et al.* [14] proposed a method to create bilingual word vectors by requiring words that are related across the two languages.

All the aforementioned approaches rely on having access to a set of correspondences between English and the target language. Some of these methods also require translating the target language to English. Yet machine translation is a very challenging task in NLP and represents an additional source of error in the classification system, due to various problems such as sparseness and noise in the data [7]. Furthermore, such methods crucially rely on accurate language identification, which is a very difficult task, especially on short texts. See e.g. [20, 19] for an overview of these methods and their limitations in generalizing to different domains.

In this work, we also investigate the performance of a language-independent classifier consisting of a CNN trained on all languages at once. This approach is similar to the Naïve Bayes classifier proposed in [25], excepts that it relies on simple hand-crafted word-level features instead of the CNN architecture used in this work.

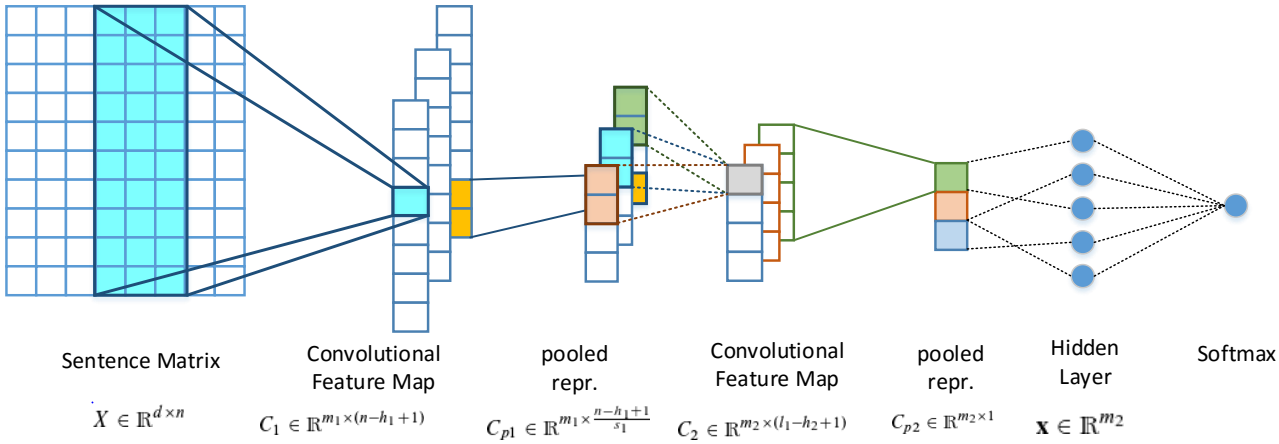


Figure 1: Architecture of the proposed CNN model with 2 convolutional layers

3. MODEL

Our model follows a multi-layer CNN architecture, which we firstly introduced in [9]. Given an input sequence of words, the corresponding sequence of word embeddings is fed as input to the first 1d convolutional layer. Each convolutional filter here operates in a sliding window fashion along the input dimension (details are described below). This layer is followed by a max-pooling operation whose output is then fed into the next convolutional layer. We extend a single-layer CNN, originally proposed in [32, 18, 17], to two convolutional and pooling layers. The resulting network architecture is illustrated in Figure 1 and in its basic variant consists of two consecutive pairs of convolutional-pooling layers followed by a single hidden layer and a soft-max output layer. In the following, we describe in detail each layer and corresponding parameters.

3.1 Convolutional Neural Network

Embedding layer. Each word is associated with a d-dimensional vector (embedding). An input sequence of n words is represented by concatenating their embeddings, yielding a sentence matrix $\mathbf{X} \in \mathbb{R}^{\mathrm{d} \times \mathrm{n}}$. $\mathbf{X}$ is used as input to the network.

Convolutional layer. This layer applies a set of m convolutional filters of length h over the matrix $\mathbf{X}$. Let $\mathbf{X}_{[i:i+h]}$ denote the concatenation of word vectors $\mathbf{x}_i$ to $\mathbf{x}_{i+h}$. A feature c_i is generated for a given filter $\mathbf{F}$ by:

$$c_i := \sum_{k,j} (\mathbf{X}_{[i:i+h]})_{k,j} \cdot \mathbf{F}_{k,j} \tag{1}$$

The concatenation of all vectors in a sentence defines a feature vector $\mathbf{c} \in \mathbb{R}^{\mathrm{n-h+1}}$. The vectors $\mathbf{c}$ are then aggregated from all m filters into a feature map matrix $\mathbf{C} \in \mathbb{R}^{\mathrm{m} \times (\mathrm{n-h+1})}$. The filters are learned during the training phase of the neural network, as described in Section 3.2. The output of the convolutional layer is passed through a non-linear activation function, before entering a pooling layer.

Pooling layer. The pooling layer aggregates the input vectors by taking the maximum over a set of non-overlapping intervals. The resulting pooled feature map matrix has the form: $\mathbf{C_{pooled}} \in \mathbb{R}^{m \times \frac{n-h+1}{s}}$, where s is the length of each interval. In the case of overlapping intervals with a stride value s_t, the pooled feature map matrix has the form $\mathbf{C_{pooled}} \in \mathbb{R}^{m \times \frac{n-h+1-s}{s_t}}$. Depending on whether the boundaries are included or not, the result of the fraction is rounded up or down respectively.

Hidden layer. A fully connected hidden layer computes the transformation $\alpha(\mathbf{W} * \mathbf{x} + \mathbf{b})$, where $\mathbf{W} \in \mathbb{R}^{\mathrm{m} \times \mathrm{m}}$ is the weight matrix, $\mathbf{b} \in \mathbb{R}^{\mathrm{m}}$ the bias, and α the rectified linear (`relu`) activation function [23]. The output $\mathbf{x} \in \mathbb{R}^{\mathrm{m}}$ of this layer can be seen as an embedding of the input sentence.

Softmax. Finally, the outputs of the previous layer $\mathbf{x} \in \mathbb{R}^{\mathrm{m}}$ are fully connected to a soft-max regression layer, which returns the class $\hat{y} \in [1, K]$ with largest probability, i.e.,

$$\begin{aligned} \hat{y} &= \arg\max_j \; P(y = j \mid \mathbf{x}, \mathbf{w}, \mathbf{a}) \\ &= \arg\max_j \; \frac{e^{\mathbf{x}^\intercal \mathbf{w}_j + a_j}}{\sum_{k=1}^{K} e^{\mathbf{x}^\intercal \mathbf{w}_k + a_j}}, \end{aligned} \tag{2}$$

where $\mathbf{w}_j$ denotes the weights vector of class j, from which the dot product with the input is formed, and a_j the bias of class j.

Network Parameters. The following parameters of the neural network are learned during training: $\theta = \{\mathbf{X}, \mathbf{F}_1, \mathbf{b}_1, \mathbf{F}_2, \mathbf{b}_2, \mathbf{W}, \mathbf{a}\}$, with $\mathbf{X}$ the word embedding matrix, where each row contains the d-dimensional embedding vector for a specific word; $\mathbf{F}_i, \mathbf{b}_i$ the filter weights and biases of convolutional layers; and $\mathbf{W}$ and $\mathbf{a}$ the weight-matrix for output classes in the soft-max layer.

3.2 Learning the Model Parameters

The model parameters are learned using the following three-phase procedure: (i) creation of word embeddings; (ii) distant-supervised phase, where the network parameters are tuned by training on weakly labelled examples; and (iii) final supervised phase, where the network is trained on the supervised training data.

Preprocessing and Word Embeddings. The word embeddings are learned on an unsupervised corpus containing $300M$ tweets. We apply a skip-gram model of window-size 5 and filter words that occur less than 15 times [32]. The dimensionality of the vector representation is set to $d = 52$. Our experiments showed that using a larger dimension did not yield any significant improvement.

Training. During the first distant-supervised phase, we use emoticons to infer noisy labels on tweets in the training set [26, 13].

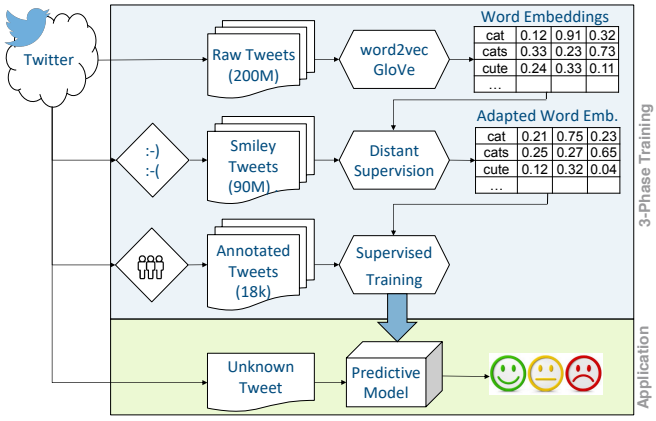


Figure 2: *Training Phases Overview.*

Table 1: *Data used for training the CNN model.*

Language	Dataset	Total	Neutral	Neg.	Pos.
English	Word embeddings	300M	-	-	-
	Pre-training	60M	-	30M	30M
	Training	18044	7544	2748	7752
	Validation	2390	987	365	1038
	Test	20632	10342	3231	7059
French	Word embeddings	300M	-	-	-
	Pre-training	60M	-	20M	40M
	Training	9107	4095	2120	2892
	Test	3238	1452	768	1018
German	Word embeddings	300M	-	-	-
	Pre-training	40M	-	8M	32M
	Training	8955	5319	1443	2193
	Test	994	567	177	250
Italian	Word embeddings	300M	-	-	-
	Pre-training	40M	-	10M	30M
	Training	6669	2942	2293	1434
	Test	741	314	250	177

Details of the training set obtained from this procedure are listed in Table 1, "Pre-training" rows. Note that the data used for this pre-training phase do not necessarily overlap with the data used to create the word embeddings. The neural network was trained on these datasets for one epoch, before finally training on the supervised data for about 20 epochs. The word embeddings $\mathbf{X} \in \mathbb{R}^{d \times n}$ are updated during both the distant- and supervised training phases by applying back-propagation through the entire network. The resulting effect on the embeddings of words are illustrated in Figure 7 and discussed in Section 5.

Optimization. During both training phases, the network parameters are learned using *AdaDelta* [36]. We compute the score on the validation set at fixed intervals and select the parameters achieving the highest score.

Figure 2 shows a complete overview of the three phases of the learning procedure.

4. EXPERIMENTAL SETTING

4.1 Data

We used a large set of $300M$ tweets to create the word embeddings for each language, as well as a distant supervision corpus of 40-60M tweets for each language, where each tweet contained at least one emoticon (positive or negative smiley). Smileys were used to automatically infer weak labels and subsequently removed from the tweets. This idea of distant-supervised learning was described in [13, 32, 31]. For the final supervised phase, we used publicly available labeled datasets for English [24], Italian [30] and French [8]. The German corpus was newly created by the authors and is available at `http://spinningbytes.com/resources`. An overview of the datasets used in our experiment, including the number of labelled tweets per dataset, is given in Table 1.

Data Preparation. Each tweet was preprocessed in three steps: (i) URLs and usernames were substituted by a replacement token, (ii) the text was lowercased and (iii) finally tokenized using the NLTK tokenizer.

4.2 Sentiment Analysis Systems

In our experiments, we compare the performance of the following sentiment analysis systems:

- *Random forest* (RF) as a common baseline classifier. The RF was trained on n-gram features, as described in [25]
- *Single-language CNN* (SL-CNN). The CNN with three-phase training, as described in Section 3, is trained for each single language. In a set of experiments, the amount of training in the three phases is gradually reduced. The system using all available training data for one language is also referred to as 'fully-trained CNN'
- *Multi-language CNN* (ML-CNN), where the distant-supervised phase is performed jointly for all languages at once, and the final supervised phase independently for each language. For the pre-training phase, we used a balanced set of $300M$ that included all four languages, see Table 1, 'Pre-training'
- *Fully multi-language CNN* (FML-CNN), where all training phases were performed without differentiation between languages. The pre-training data is the same as in ML-CNN
- *SemEval benchmark.* In addition, results on the English dataset were compared to the best known ones from the SemEval benchmark[1]. For the data sets in the other three languages, no public benchmark results could be found in literature
- *Translate*: this approach uses Google Translate[2] (as of Oct 2016) to translate each input text from a source language to a target language. It then uses the SL-CNN classifier trained for the target language to classify the tweets

4.3 Performance Measure

We evaluate the performance of the proposed models using the metric of SemEval-2016 challenge, which consists in averaging the macro F1-score of the positive and negative classes[3]. Each approach was trained for a fixed number of epochs. We then selected the results that yielded the best results on a separate validation set.

For French, German and Italian, we created a validation set by randomly sampling 10% of the data. For English we used the `test2015` set as validation set and `test2016` for testing from the SemEval-2016 challenge, see Validation set in Table 1.

[1] `http://alt.qcri.org/semeval2016/`
[2] `https://translate.google.com/`
[3] Note that this still takes into account the prediction preformance on the neutral class.

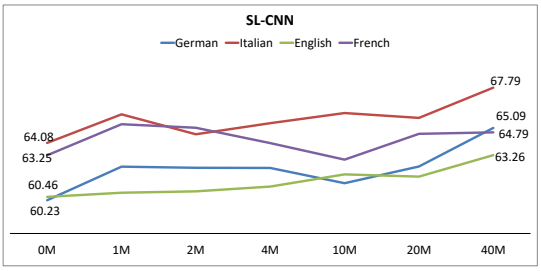


Figure 3: *Results obtained by varying the amount of data during the distant supervised phase.* Each CNN was trained for one epoch.

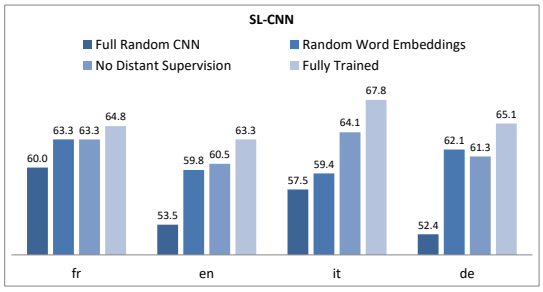


Figure 4: *Results obtained by initializing the CNNs with different word embeddings.* The fully trained variant typically performs better than the other three variants, thus demonstrating the importance of initializing the words vectors as well as performing distant-supervised training.

4.4 Implementation Details

The core routines of our system are written in `Theano` [3] exploiting GPU acceleration with the `CuDNN` library [5]. The whole learning procedure takes approximately 24-48 hours to create the word embeddings, 20 hours for the distant-supervised phase with $160M$ tweets and only 30 minutes for the supervised phase with $35K$ tweets.

Experiments were conducted on 'g2.2xlarge' instances of *Amazon Web Services* (AWS) with *GRID K520* GPU having 3072 CUDA cores and 8 GB of RAM.

5. RESULTS

In this section, we summarize the main results of our experiments.

The F1-scores of the proposed approach and competing baselines are summarized in Table 2. The fully-trained SL-CNNs significantly outperforms the other methods in all four languages. The best F1-score was achieved for Italian (67.79%), followed by German (65.09%) and French (64.79%), while the system for English reached only 62.26%. The proposed SL-CNNs outperform the corresponding baselines from literature and RF.

Leveraging Distant Training Data. We increased the amount of data for the distant-supervised phase for SL-CNN. Figure 3 compares F1-scores for each language when changing the amount of tweets from 0 to 40M. The scores without distant supervision are the lowest for all languages. We observe a general increase of F1-score when increasing the amount of training data. The performance gain for English, Italian and German is around 3%, while it is more moderate for French.

Table 2: *F1-scores of compared methods on the test sets.* The highest scores among the three proposed models are highlighted in bold face. **ML-CNN** and **FML-CNN** are two variants of the method presented in Section 5.1.

Method	**Language**			
	English	**French**	**German**	**Italian**
SL-CNN	**63.49**	**64.79**	**65.09**	**67.79**
ML-CNN	61.61	-	63.62	64.73
FML-CNN	61.03	-	63.19	64.80
RF	48.60	53.86	52.40	52.71
SENSEI-LIF [24]	62.96	-	-	-
UNIMELB [24]	61.67	-	-	-

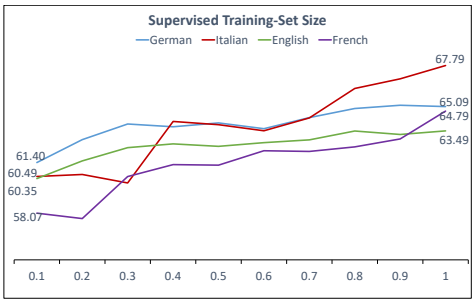


Figure 5: *Results obtained by varying the amount of supervised data. The maximum number on the y-axis corresponds to the total number of training tweets given in Table 1.*

Supervised data. In Figure 5 we report the F1-scores of each model for increasing amount of supervised data. We observe a score increase of 2-4% when using 100% of the available data instead of 10%.

Word Embeddings. We investigate the importance of initialization of word embeddings and the interaction of the latter with the distant supervised phase in four scenarios: (i) using randomly initialized word embedding weights, not updated during the distant-supervised phase (named *Full Random CNN*), (ii) using randomly initialized word embeddings, updated during the distant-supervised phase (*Random Word Embeddings*), (iii) using word2vec embeddings without distant supervision (*No Distant Supervision*) and (iv) using word2vec embeddings, updated during the distant-supervised phase using $160M$ tweets (*Fully trained CNN*). Results in Figure 4 demonstrate that the *Fully trained CNN* approach performed the best in almost all cases. These results prove that the quality of initialization as well as updating the large number of word vector parameters during training of the network yield significant improvements.

Figure 7 illustrates the effect of the distant-supervised phase on the word embeddings. For visualization purposes, principal component analysis (PCA) was used to project the word embeddings onto two dimensions. We see that the geometry of the word embeddings reflects the distance in terms of sentiment between pairs of words. Figure 7(a) shows the initial word embeddings created by word2vec, before the distant-supervised phase. Taking as an ex-

Table 3: *Summary of the network parameters used for experimental results.* L2 is the default architecture used by our approach. The alternative L1 and L3 architectures are discussed in Figure 8.

	Number of layers	Number of filters	Filter window size h	Size of max-pooling window w and striding st
L1	1	300	$h_1 = 5$	None
L2	2	200	$h_1 = 4, h_2 = 3$	$w_1 = 4, st_1 = 2$
L3	3	200	$h_1 = 4, h_2 = 3, h_3 = 2$	$w_1 = 4, st_1 = 2, w_2 = 3, st_2 = 1$

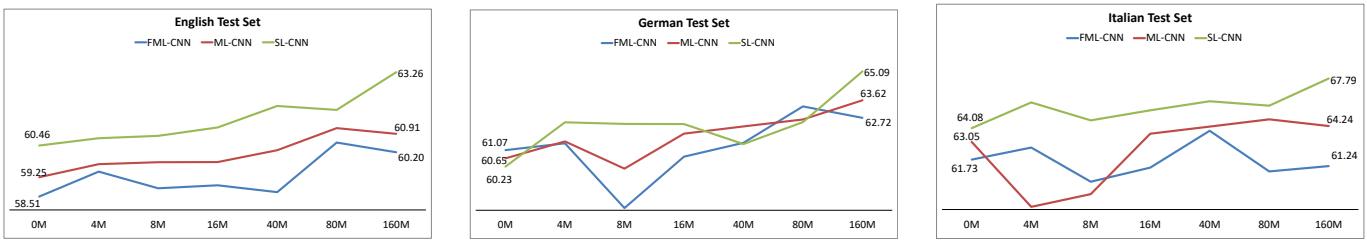
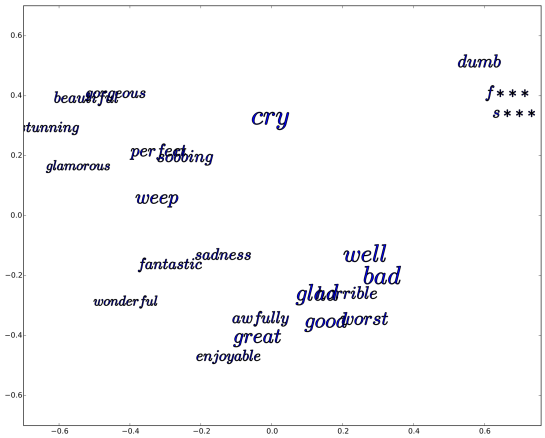
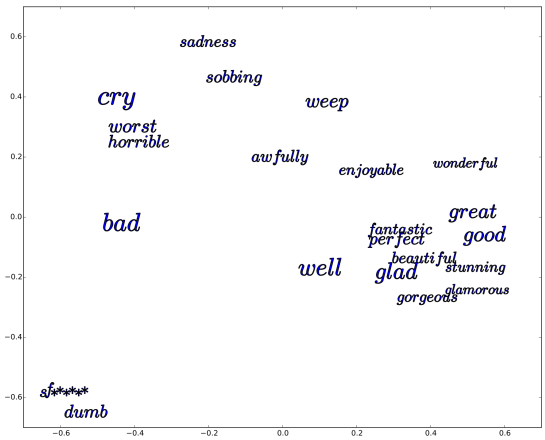


Figure 6: *Results obtained by varying the amount of data during the distant supervised phase.* Each CNN was trained for one epoch. We rescaled the curve for *SL-CNN* to match the amount of data used per language by the multi-language approaches. For example, while the multi-language approaches were trained with $40M$ tweets ($10M$ tweets per language), each *SL-CNN* model was trained with $10M$ tweets from the same language. Each experiment set, up to $160M$ tweets, contains the same number of tweets per language.



(a)



(b)

Figure 7: *Word embeddings projected onto a 2-dimensional space using PCA before (a) and after (b) the distant-supervised training phase.*

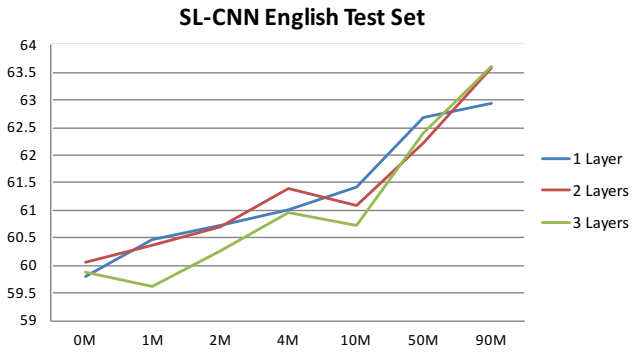


Figure 8: *F1-score on the English corpus as a function of the number of network layers.* These results were obtained by training the *SL-CNN* models with different number of layers, as specified in Table 3, on different amounts of data during the distant-supervised phase. Each CNN was trained for one distant epoch.

ample the pair of words "good" and "bad", it is clear that these two words often appear in the same context and are thus close to each other in the embedded space. The similarity score of these two vectors is 0.785. After the distant-supervised phase, the semantic of the space is changed and the distance between words come to reflect the difference in terms of sentiment. As shown in Figure 7(b), negative and positive words are neatly separated into two clusters. In this case, the similarity score between the word embeddings of "good" and "bad" becomes -0.055. Finer grained clusters are also revealed in the second embedding. For example, words that convey sadness are close together.

Comparing Network Architectures. One common question asked by practitioners relates to the influence of the number of layers on the performance of a neural network. We thus evaluated the performance of various architectures with multiple layers. In order to reduce the number of experiments, we evaluated the per-

Table 4: *Translation experiment.* We translated each source language to each target language and used the model trained on the target language to predict tweets polarity.

Source	Target			
	English	Italian	German	French
English	**63.49**	59.87	57.53	58.47
Italian	64.37	**67.79**	61.57	60.19
German	61.86	61.66	**65.09**	61.22
French	**65.68**	63.23	61.5	64.79

formance of the single-language model SL-CNN on the English set and varied the number of convolutional/pooling layer pairs from 1 to 3. We evaluated a total of 12 networks. Here, we only report the best set of parameters for each number of layers in Table 3 and corresponding F1-scores in Figure 8. The network performance generally improves with the number of layers, if a sufficient amount of training data is used in the distant-supervised phase. For the task of sentiment classification, current recurrent architectures such at LSTMs still do not perform as well as CNNs, see e.g. [29, Table 4] and the discussion in the related work section.

Translation Approach. In Table 4 we report results of the translation experiment described in Section 4.2. The F1-score is higher when not translating tweets to another language for English, Italian and German. As an exception, we obtained better results when translating French to English and using the English model to predict sentiments.

5.1 Comparison to multi-language classifiers

Figure 6 summarizes F1-scores of the three CNN variants described in Section 4.2, namely *SL-*, *ML-* and *FML-CNN*, when varying the amount of distant-supervised phase. When comparing the three CNN variants, we see that *SL-CNN* gets slightly better scores than *ML-CNN* and *FML-CNN*. The difference in performance between the single and multi-language models is around 2% on average. However, one benefit of the multi-language models over the single-language ones is their ability to deal with text in mixed languages. To check this hypothesis, we used the *langpi* tool [20] to extract a set of 300 tweets from the German corpus containing English words. Although these tweets were classified by Twitter as German, they contain a significant number of English words (some of them entirely written in English). We also manually inspected this set and discarded tweets that did not contain English. We then retrained the two models on the training set from which we first removed the set of 300 tweets. When evaluating on this subset, *ML-CNN* obtained an F1-score of 68.26 while *SL-CNN* obtained 64.07. When manually inspecting the results, we clearly observed that *ML-CNN* was better at classifying tweets that were entirely in English or contained several English words. The effect of using different word embedding initializations in the multilingual networks is summarized in Figure 9.

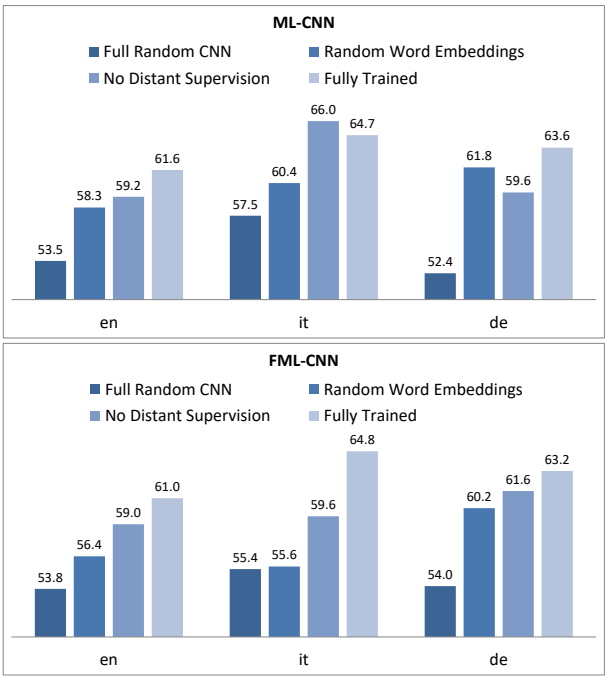


Figure 9: *Results obtained by initializing the CNNs with different word embeddings.* As for the results obtained with the *SL-CNN* model, the fully trained variant typically performs better than the other three variants, both for the *ML-CNN* and *FML-CNN* models.

6. CONCLUSION

We described a deep learning framework to predict the sentiment polarity of short texts written in multiple languages. In contrast to most existing methods, our approach does not rely on establishing a correspondence to English but instead exploits large amounts of weakly supervised data to train a multi-layer CNN directly in the target language. Through a thorough experimental evaluation, we addressed some fundamental questions around the performance of such model. First, we demonstrated that the strategy used to train these models plays an important role in the obtained performance. Two important factors are a good initialization for the word vectors as well as pre-training using large amounts of weakly supervised data. Second, we compared the performance of a single-language and a multi-language approach. The single-language model reaches the best performance and it even outperforms existing state-of-the-art methods on all the datasets of the SemEval-2016 competition. The multi-language approach performs comparably well or slightly worse than its single-language counterpart, while exhibiting several advantages: it does not need to know *a priori* the language(s) used in each tweet; the model can be easily extended to more languages; and it can cope with texts written in multiple languages.

Acknowledgments.

This research has been funded by Commission for Technology and Innovation (CTI) project no. 18832.1 PFES-ES, and by SpinningBytes AG, Switzerland.

7. REFERENCES

[1] M. Araujo, J. Reis, A. Pereira, and F. Benevenuto. An evaluation of machine translation for multilingual sentence-level sentiment analysis. In *Proceedings of the 31st Annual ACM Symposium on Applied Computing*, pages 1140–1145, 2016.

[2] A. Balahur and M. Turchi. Multilingual sentiment analysis using machine translation? In *Proceedings of the 3rd Workshop in Computational Approaches to Subjectivity and Sentiment Analysis*, 2012.

[3] J. Bergstra, O. Breuleux, F. F. Bastien, P. Lamblin, R. Pascanu, G. Desjardins, J. Turian, D. Warde-Farley, and Y. Bengio. Theano: a CPU and GPU math compiler in Python. *Proceedings of the Python for Scientific Computing Conference (SciPy)*, pages 1–7, 2010.

[4] E. Boiy and M.-F. Moens. A machine learning approach to sentiment analysis in multilingual web texts. *Information retrieval*, 12(5):526–558, 2009.

[5] S. Chetlur and C. Woolley. cuDNN: Efficient Primitives for Deep Learning. *arXiv preprint*, pages 1–9, 2014.

[6] M. Cieliebak, O. Dürr, and F. Uzdilli. Potential and limitations of commercial sentiment detection tools. In *ESSEM@ AI* IA*, pages 47–58, 2013.

[7] K. Dashtipour, S. Poria, A. Hussain, E. Cambria, A. Y. A. Hawalah, A. Gelbukh, and Q. Zhou. Multilingual sentiment analysis: State of the art and independent comparison of techniques. *Cognitive Computation*, 2016.

[8] DEFT. `https://deft.limsi.fr/2015/`, 2015.

[9] J. Deriu, M. Gonzenbach, F. Uzdilli, A. Lucchi, V. De Luca, and M. Jaggi. Swisscheese at semeval-2016 task 4: Sentiment classification using an ensemble of convolutional neural networks with distant supervision. *Proceedings of SemEval*, pages 1124–1128, 2016.

[10] C. N. dos Santos and M. Gatti. Deep Convolutional Neural Networks for Sentiment Analysis of Short Texts. In *COLING*, pages 69–78, 2014.

[11] D. Erhan, Y. Bengio, A. Courville, P.-A. Manzagol, P. Vincent, and S. Bengio. Why does unsupervised pre-training help deep learning? *The Journal of Machine Learning Research*, 11:625–660, 2010.

[12] J. Gao, X. He, W.-t. Yih, and L. Deng. Learning continuous phrase representations for translation modeling. In *ACL*, 2014.

[13] A. Go, R. Bhayani, and L. Huang. Twitter Sentiment Classification using Distant Supervision. Technical report, The Stanford Natural Language Processing Group, 2009.

[14] S. Gouws, Y. Bengio, and G. Corrado. Bilbowa: Fast bilingual distributed representations without word alignments. In *Proceedings of the 32nd International Conference on Machine Learning*, 2015.

[15] IWS. `http://www.internetworldstats.com/stats7.htm`, 2016.

[16] R. Johnson and T. Zhang. Semi-supervised Convolutional Neural Networks for Text Categorization via Region Embedding. In *Advances in Neural Information Processing Systems 28*, pages 919–927, 2015.

[17] N. Kalchbrenner, E. Grefenstette, and P. Blunsom. A Convolutional Neural Network for Modelling Sentences. In *ACL - Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 655–665, 2014.

[18] Y. Kim. Convolutional Neural Networks for Sentence Classification. In *EMNLP 2014 - Empirical Methods in Natural Language Processing*, pages 1746–1751, 2014.

[19] M. Lui and T. Baldwin. Cross-domain feature selection for language identification. In *In Proceedings of 5th International Joint Conference on Natural Language Processing*, 2011.

[20] M. Lui and T. Baldwin. langid. py: An off-the-shelf language identification tool. In *Proceedings of the ACL 2012 system demonstrations*, pages 25–30. Association for Computational Linguistics, 2012.

[21] Mashable. `http://mashable.com/2013/12/17/twitter-popular-languages/#gaFEjWnHPkql`, 2013.

[22] R. Mihalcea, C. Banea, and J. M. Wiebe. Learning multilingual subjective language via cross-lingual projections. In *ACL*, 2007.

[23] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning*, pages 807–814, 2010.

[24] 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*, SemEval '16, June 2016.

[25] S. Narr, M. Hulfenhaus, and S. Albayrak. Language-independent twitter sentiment analysis. *Knowledge Discovery and Machine Learning (KDML), LWA*, pages 12–14, 2012.

[26] J. Read. Using emoticons to reduce dependency in machine learning techniques for sentiment classification. In *Proceedings of the ACL student research workshop*, pages 43–48. Association for Computational Linguistics, 2005.

[27] F. N. Ribeiro, M. Araújo, P. Gonçalves, F. Benevenuto, and M. A. Gonçalves. A benchmark comparison of state-of-the-practice sentiment analysis methods. *arXiv preprint arXiv:1512.01818*, 2015.

[28] S. Rothe, S. Ebert, and H. Schutze. Ultradense Word Embeddings by Orthogonal Transformation. *arXiv*, 2016.

[29] S. Semeniuta, A. Severyn, and E. Barth. Recurrent Dropout without Memory Loss. *arXiv*, 2016.

[30] sentipolc. `http://www.di.unito.it/~tutreeb/sentipolc-evalita16/data.html`, 2016.

[31] A. Severyn and A. Moschitti. Twitter Sentiment Analysis with Deep Convolutional Neural Networks. In *38th International ACM SIGIR Conference*, pages 959–962, 2015.

[32] A. Severyn and A. Moschitti. UNITN: Training Deep Convolutional Neural Network for Twitter Sentiment Classification. In *SemEval 2015 - Proceedings of the 9th International Workshop on Semantic Evaluation*, 2015.

[33] Y. Shen, X. He, J. Gao, L. Deng, and G. Mesnil. A latent semantic model with convolutional-pooling structure for information retrieval. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, pages 101–110, 2014.

[34] R. Socher, A. Perelygin, J. Y. Wu, J. Chuang, C. D. Manning, A. Y. Ng, and C. Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the conference on empirical methods in natural language processing*, volume 1631, page 1642, 2013.

[35] M. Wick, P. Kanani, and A. Pocock. Minimally-constrained multilingual embeddings via artificial code-switching. 2015.

[36] M. D. Zeiler. ADADELTA: An Adaptive Learning Rate Method. *arXiv*, 2012.