Learning Representations for Tweets through Word Embeddings

Cedric De Boom, Steven Van Canneyt, Thomas Demeester,
Bart Dhoedt FIRSTNAME.LASTNAME@UGENT.BE
Ghent University – iMinds, INTEC, Technologiepark 15, 9052 Zwijnaarde, Belgium

Abstract

In event detection and opinion mining on social media it is important to grasp the semantic meaning of a text post. In this abstract, we present a method to learn effective representations for Twitter posts through a combination of word embeddings and word frequency information. We design a semantic similarity task between tweet couples and a novel loss function to train our model. We test it on a manually crafted dataset of tweets, and we find that our method outperforms the traditional baselines.

1. Introduction

Short text messages such as tweets are very noisy and sparse in their use of vocabulary. Traditional textual representations such as tf-idf rely on exact word overlap, and therefore have difficulty grasping the semantic meaning of tweets, which is important in applications such as event detection, opinion mining, news recommendation, etc. We therefore construct a novel representation learning scheme using semantic word embeddings. Such embeddings are distributed vector representations of single words in a fixed-dimensional semantic space, as opposed to tf-idf vectors, in which a word is represented by a one-hot vector. To arrive at a text representation, word embeddings are often aggregated through a mean, max, min… function, usually in combination with a (convolutional) neural network or other classification scheme (Weston et al., 2014)(dos Santos & Gatti, 2014). Word order is lost in this case; but, on the other hand, aggregation is simple, out-of-the-box and does not require a fixed length input. In our method we will aggregate word embeddings using tf-idf information to arrive at an overall tweet representation.

The obtained tweet representations will be evaluated through a semantic similarity task. At the most strict level, semantic similarity between two texts is often defined as one being a (near) paraphrase of the other. In our, more relaxed, setting we are interested in topic- and subject-related texts, as this is often applicable in the already mentioned use cases of event detection and recommendation. For example, Van Gogh and sunflowers are topic-related, although the concepts are dissimilar in the most strict sense.

2. Methodology

The core principle is to assign a weight to each word in a tweet. These weights are determined based on the idf value of the individual words. The idea is that important words – i.e. words that are needed to determine most of the tweet's semantics – usually have higher idf values than less important words, such as articles and auxiliaries. Indeed, the latter appear more frequently in various different tweets, while words with a high idf value mostly occur in similar contexts. The final goal is to combine the weighted words into a semantically effective tweet representation.

To achieve this goal, we model this problem as a semantic similarity task. After calculating the distance between two tweet representations, we use a simple learned threshold function to determine whether the tweets are semantically related or not.

In the learning scheme, we use related and non-related tweets as input data. First, the words in a tweet are sorted from high to low idf values. After that, every embedding vector for each of the sorted words is multiplied with a weight to be learned. Finally, the weighted vectors are averaged to arrive at a single tweet representation. To learn word weights for texts with arbitrary length, we interpolate the weights linearly.

2.1. Learning procedure

Our model is related to the siamese neural network with parameter sharing (Bromley et al., 1993). We first calculate vector representations for a couple of tweets C^α and C^β, after which we compare the two vector representations $t(C^\alpha)$ and $t(C^\beta)$ through a loss $\mathcal{L}\left(t(C^\alpha), t(C^\beta)\right)$ that we wish to minimize. We then update the weights using minibatch gradient descent.

The loss function needs to express that semantically related tweets have to lie close to each other in the representation space, and non-related tweets to lie far apart from each other. A first loss function is related to the contrastive loss regularly used in siamese neural networks ((Hadsell et al., 2006)):

$$\mathcal{L}_\mathrm{c}\left(t(C^\alpha), t(C^\beta)\right) = p_C \cdot d\left(t(C^\alpha), t(C^\beta)\right),$$

in which $d(\cdot)$ is a chosen distance function and p_C is a parameter that is 1 if the tweet couple is semantically related, and -1 otherwise. This particular loss function is simple and convenient, but also has some issues. First, there is an imbalance between the loss for related pairs and non-related pairs, in which the latter can get an arbitrarily large negative loss, while the related pairs' loss cannot be pushed below zero. Second, this loss function can skew distance distributions, so that minimizing overlap between distributions is not guaranteed. In a second loss function we try to mitigate these issues using the median, as it is a very robust statistic insensitive to outliers:

$$\begin{aligned}&\mathcal{L}_\mathrm{m}\left(t(C^\alpha), t(C^\beta), B\right)\\ &= \ln\left[1 + \exp\left(-\kappa p_C\Big(\mu(B) - d\big(t(C^\alpha), t(C^\beta)\big)\Big)\right)\right],\end{aligned}$$

in which B represents a minibatch, $\mu(B)$ the median distance in that batch, and κ is a hyperparameter.

3. Data collection

We propose that two tweets are semantically related if they are generated by the same event. As in (De Boom et al., 2015b), we require that such an event is represented by one or more hashtags. Since Twitter posts and associated events are very noisy by nature, we restrict ourselves to tweets by 100 English news agencies, and we gather 341 949 tweets in total over a two-week period. We employ four heuristics to gather semantically related tweets: 1. Word count (no hashtags, mentions or URLs) ≥ 5; 2. Jaccard similarity between hashtags in both tweets ≥ 0.5; 3. Time difference $\leq$ 15 minutes; 4. Jaccard similarity between words in both tweets ≤ 0.5 (sufficiently different words in both tweets). We manually label 200 generated pairs and non-pairs, and we achieve an error rate of 28%. Achieving an error rate lower than around 28% on this dataset will therefore be difficult, and the gain would not lead to a better model of the human notion of similarity anyway.

Table 1. Results on the collected Twitter data.

	Split error	JS divergence
Tf-idf	43.09%	0.0634
Mean	33.68%	0.0783
Max	34.85%	0.0668
Min/max	33.80%	0.0734
Mean, top 30% idf	32.60%	0.0811
Max, top 30% idf	33.38%	0.0740
Min/max, top 30% idf	32.86%	0.0762
Mean, idf-weighed	31.28%	0.0886
Learned weights, $\mathcal{L}_\mathrm{c}$	35.48%	0.0658
Learned weights, $\mathcal{L}_\mathrm{m}$	**30.88%**	**0.0900**

4. Experiments

We use Google's *word2vec* software to calculate word embeddings of 400 dimensions using the cleaned English Wikipedia dump of March 4, 2015, for which we also calculate idf values. We create a train set of 30,000, a validation set of 40,000 and a test set of 27,000 tweet couples. The validation set is used to calculate the distance threshold between related and non-related tweets. As shown in (De Boom et al., 2015a), Euclidean distance is the best choice to measure semantic relatedness, so we will use it throughout the experiments. We report performance in terms of split error (should be low) and Jensen-Shannon divergence (should be high).

In Table 1 we report several baselines, such as plain tf-idf and simple aggregation of word embeddings through mean, max and a concatenation of min and max vectors. We do the same for the top 30% idf words in the tweets, and we also weigh every word directly with its idf value after which we average.

The split error remains slightly higher than the human error rate of 28%, as expected. Tf-idf performs bad in this experiment, and is clearly not fit to represent tweets efficiently. The other baselines have a much better, but overall comparable performance. Our method with median-based loss performs the best, but the contrastive loss suffers from the afore-mentioned issues.

5. Conclusion

We devised an effective method to derive vector representations for tweets using a combination of weighted word embeddings and idf values. For this purpose we designed a contrastive-based and a novel median-based loss function that can effectively mitigate the effect of outliers.

Acknowledgements

This work is published in Pattern Recognition Letters on June 28, 2016 (De Boom et al., 2016). We acknowledge Nvidia for its generous hardware support. Cedric De Boom is funded by a Ph.D. grant of the Flanders Research Foundation (FWO). Steven Van Canneyt is funded by a Ph.D. grant of the Agency for Innovation by Science and Technology in Flanders (IWT). All code for this paper is readily available on our GitHub page `github.com/cedricdeboom/RepresentationLearning`.

References

Bromley, J., Guyon, I., Lecun, Y., Säckinger, E., & Shah, R. (1993). Signature Verification Using a Siamese Time Delay Neural Network. *NIPS*.

De Boom, C., Van Canneyt, S., Bohez, S., Demeester, T., & Dhoedt, B. (2015a). Learning Semantic Similarity for Very Short Texts. *International Conference on Data Mining Workshop*.

De Boom, C., Van Canneyt, S., & Dhoedt, B. (2015b). Semantics-driven Event Clustering in Twitter Feeds. *#Microposts*.

De Boom, C., Van Canneyt, S., Demeester, T., & Dhoedt, B. (2016). Representation learning for very short texts using weighted word embedding aggregation. *Pattern Recognition Letters*.

dos Santos, C. N., & Gatti, M. (2014). Deep Convolutional Neural Networks for Sentiment Analysis of Short Texts. *COLING*.

Hadsell, R., Chopra, S., & Lecun, Y. (2006). Dimensionality Reduction by Learning an Invariant Mapping. *CVPR*.

Weston, J., Chopra, S., & Adams, K. (2014). #TagSpace: Semantic embeddings from hashtags. *EMNLP*.