Tweet2Vec: Character-Based Distributed Representations for Social Media

Bhuwan Dhingra[1], Zhong Zhou[2], Dylan Fitzpatrick[1,2]
Michael Muehl[1] and **William W. Cohen[1]**
[1]School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA
[2]Heinz College, Carnegie Mellon University, Pittsburgh, PA, USA
{bdhingra,djfitzpa,mmuehl}@andrew.cmu.edu
zhongzhou@cmu.edu wcohen@cs.cmu.edu

Abstract

Text from social media provides a set of challenges that can cause traditional NLP approaches to fail. Informal language, spelling errors, abbreviations, and special characters are all commonplace in these posts, leading to a prohibitively large vocabulary size for word-level approaches. We propose a character composition model, `tweet2vec`, which finds vector-space representations of whole tweets by learning complex, non-local dependencies in character sequences. The proposed model outperforms a word-level baseline at predicting user-annotated *hashtags* associated with the posts, doing significantly better when the input contains many out-of-vocabulary words or unusual character sequences. Our *tweet2vec* encoder is publicly available[1].

1 Introduction

We understand from Zipf's Law that in any natural language corpus a majority of the vocabulary word types will either be absent or occur in low frequency. Estimating the statistical properties of these rare word types is naturally a difficult task. This is analogous to the curse of dimensionality when we deal with sequences of tokens - most sequences will occur only once in the training data. Neural network architectures overcome this problem by defining non-linear compositional models over vector space representations of tokens and hence assign non-zero probability even to sequences not seen during training (Bengio et al., 2003; Kiros et al., 2015). In this work, we explore a similar approach to learning distributed representations of social media posts by composing them from their constituent *characters*, with the goal of generalizing to out-of-vocabulary words as well as sequences at test time.

Traditional Neural Network Language Models (NNLMs) treat words as the basic units of language and assign independent vectors to each word type. To constrain memory requirements, the vocabulary size is fixed before-hand; therefore, rare and out-of-vocabulary words are all grouped together under a common type 'UNKNOWN'. This choice is motivated by the assumption of arbitrariness in language, which means that surface forms of words have little to do with their semantic roles. Recently, (Ling et al., 2015) challenge this assumption and present a bidirectional Long Short Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) for composing word vectors from their constituent characters which can memorize the arbitrary aspects of word orthography as well as generalize to rare and out-of-vocabulary words.

Encouraged by their findings, we extend their approach to a much larger unicode character set, and model long sequences of text as functions of their constituent characters (including whitespace). We focus on social media posts from the website Twitter, which are an excellent testing ground for character based models due to the noisy nature of text. Heavy use of slang and abundant misspellings means that there are many orthographically and semantically similar tokens, and special characters such as emojis are also immensely popular and carry useful semantic information. In our moderately sized training dataset of 2 million tweets, there were about 0.92 million unique word types. It would be expensive to capture all these phenomena in a word based model in terms of both the memory requirement (for the increased vocabulary) and the amount of training data required for effective learning. Additional benefits of the character based approach

[1]https://github.com/bdhingra/tweet2vec

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 269–274,
Berlin, Germany, August 7-12, 2016.

include language independence of the methods, and no requirement of NLP preprocessing such as word-segmentation.

A crucial step in learning good text representations is to choose an appropriate objective function to optimize. Unsupervised approaches attempt to reconstruct the original text from its latent representation (Mikolov et al., 2013; Bengio et al., 2003). Social media posts however, come with their own form of supervision annotated by millions of users, in the form of *hashtags* which link posts about the same topic together. A natural assumption is that the posts with the same hashtags should have embeddings which are close to each other. Hence, we formulate our training objective to maximize cross-entropy loss at the task of predicting hashtags for a post from its latent representation.

We propose a Bi-directional Gated Recurrent Unit (Bi-GRU) (Chung et al., 2014) neural network for learning tweet representations. Treating white-space as a special character itself, the model does a forward and backward pass over the entire sequence, and the final GRU states are linearly combined to get the tweet embedding. Posterior probabilities over hashtags are computed by projecting this embedding to a softmax output layer. Compared to a word-level baseline this model shows improved performance at predicting hashtags for a held-out set of posts. Inspired by recent work in learning vector space text representations, we name our model *tweet2vec*.

2 Related Work

Using neural networks to learn distributed representations of words dates back to (Bengio et al., 2003). More recently, (Mikolov et al., 2013) released `word2vec` - a collection of word vectors trained using a recurrent neural network. These word vectors are in widespread use in the NLP community, and the original work has since been extended to sentences (Kiros et al., 2015), documents and paragraphs (Le and Mikolov, 2014), topics (Niu and Dai, 2015) and queries (Grbovic et al., 2015). All these methods require storing an extremely large table of vectors for all word types and cannot be easily generalized to unseen words at test time (Ling et al., 2015). They also require preprocessing to find word boundaries which is non-trivial for a social network domain like Twitter.

In (Ling et al., 2015), the authors present a compositional character model based on bidirectional LSTMs as a potential solution to these problems. A major benefit of this approach is that large word lookup tables can be compacted into character lookup tables and the compositional model scales to large data sets better than other state-of-the-art approaches. While (Ling et al., 2015) generate word embeddings from character representations, we propose to generate vector representations of entire tweets from characters in our `tweet2vec` model.

Our work adds to the growing body of work showing the applicability of character models for a variety of NLP tasks such as Named Entity Recognition (Santos and Guimarães, 2015), POS tagging (Santos and Zadrozny, 2014), text classification (Zhang et al., 2015) and language modeling (Karpathy et al., 2015; Kim et al., 2015).

Previously, (Luong et al., 2013) dealt with the problem of estimating rare word representations by building them from their constituent morphemes. While they show improved performance over word-based models, their approach requires a morpheme parser for preprocessing which may not perform well on noisy text like Twitter. Also the space of all morphemes, though smaller than the space of all words, is still large enough that modelling all morphemes is impractical.

Hashtag prediction for social media has been addressed earlier, for example in (Weston et al., 2014; Godin et al., 2013). (Weston et al., 2014) also use a neural architecture, but compose text embeddings from a lookup table of words. They also show that the learned embeddings can generalize to an unrelated task of document recommendation, justifying the use of hashtags as supervision for learning text representations.

3 Tweet2Vec

Bi-GRU Encoder: Figure 1 shows our model for encoding tweets. It uses a similar structure to the C2W model in (Ling et al., 2015), with LSTM units replaced with GRU units.

The input to the network is defined by an alphabet of characters C (this may include the entire unicode character set). The input tweet is broken into a stream of characters $c_1, c_2, ...c_m$ each of which is represented by a 1-by-$|C|$ encoding. These one-hot vectors are then projected to a character space by multiplying with the matrix $P_C \in$

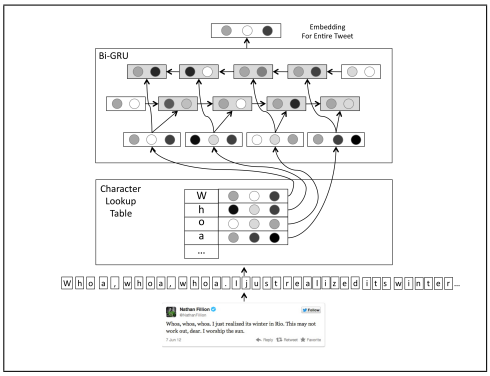


Figure 1: Tweet2Vec encoder for social media text

$\mathbb{R}^{|C| \times d_c}$, where d_c is the dimension of the character vector space. Let $x_1, x_2, ...x_m$ be the sequence of character vectors for the input tweet after the lookup. The encoder consists of a forward-GRU and a backward-GRU. Both have the same architecture, except the backward-GRU processes the sequence in reverse order. Each of the GRU units process these vectors sequentially, and starting with the initial state h_0 compute the sequence $h_1, h_2, ...h_m$ as follows:

$$
\begin{aligned}
r_t &= \sigma(W_r x_t + U_r h_{t-1} + b_r), \\
z_t &= \sigma(W_z x_t + U_z h_{t-1} + b_z), \\
\tilde{h}_t &= tanh(W_h x_t + U_h(r_t \odot h_{t-1}) + b_h), \\
h_t &= (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t.
\end{aligned}
$$

Here r_t, z_t are called the *reset* and *update* gates respectively, and $\tilde{h}_t$ is the *candidate* output state which is converted to the actual output state h_t. W_r, W_z, W_h are $d_h \times d_c$ matrices and U_r, U_z, U_h are $d_h \times d_h$ matrices, where d_h is the hidden state dimension of the GRU. The final states h_m^f from the forward-GRU, and h_0^b from the backward GRU are combined using a fully-connected layer to the give the final tweet embedding e_t:

$$e_t = W^f h_m^f + W^b h_0^b \tag{1}$$

Here W^f, W^b are $d_t \times d_h$ and b is $d_t \times 1$ bias term, where d_t is the dimension of the final tweet embedding. In our experiments we set $d_t = d_h$. All parameters are learned using gradient descent.

Softmax: Finally, the tweet embedding is passed through a linear layer whose output is the same size as the number of hashtags L in the data set. We use a softmax layer to compute the posterior hashtag probabilities:

$$P(y = j|e) = \frac{exp(w_j^T e + b_j)}{\sum_{i=1}^{L} exp(w_i^T e + b_j)}. \tag{2}$$

Objective Function: We optimize the categorical cross-entropy loss between predicted and true hashtags:

$$J = \frac{1}{B} \sum_{i=1}^{B} \sum_{j=1}^{L} -t_{i,j} log(p_{i,j}) + \lambda \|\Theta\|^2. \tag{3}$$

Here B is the batch size, L is the number of classes, $p_{i,j}$ is the predicted probability that the i-th tweet has hashtag j, and $t_{i,j} \in \{0, 1\}$ denotes the ground truth of whether the j-th hashtag is in the i-th tweet. We use L2-regularization weighted by λ.

4 Experiments and Results

4.1 Word Level Baseline

Since our objective is to compare character-based and word-based approaches, we have also implemented a simple word-level encoder for tweets. The input tweet is first split into tokens along white-spaces. A more sophisticated tokenizer may be used, but for a fair comparison we wanted to keep language specific preprocessing to a minimum. The encoder is essentially the same as *tweet2vec*, with the input as words instead of characters. A lookup table stores word vectors for the V (20K here) most common words, and the rest are grouped together under the 'UNK' token.

4.2 Data

Our dataset consists of a large collection of global posts from Twitter[2] between the dates of June 1, 2013 to June 5, 2013. Only English language posts (as detected by the *lang* field in Twitter API) and posts with at least one hashtag are retained. We removed infrequent hashtags (< 500 posts) since they do not have enough data for good generalization. We also removed very frequent tags ($> 19K$ posts) which were almost always from automatically generated posts (ex: `#androidgame`) which are trivial to predict. The final dataset contains 2 million tweets for training, 10K for validation and 50K for testing, with a total of 2039 distinct hashtags. We use simple regex to preprocess the post text and remove hashtags (since these are to be predicted) and HTML tags, and replace usernames and URLs with special tokens. We also removed *retweets* and convert the text to lower-case.

[2] `https://twitter.com/`

Tweets	Word model baseline	*tweet2vec*
ninety-one degrees. ☀❤😇	#initialsofsomeone.. #nw #gameofthrones	#summer **#loveit** #sun
self-cooked scramble egg. yum!! !url	#music #cheap #cute	**#yummy** #food #foodporn
can't sleeeeeeep	#gameofthrones #heartbreaker	#tired **#insomnia**
oklahoma!!!!!!!!!! champions!!!!!	#initialsofsomeone.. #nw #lrt	**#wcws** #sooners #ou
7 % of battery . iphones die too quick .	#help #power #money #s	#fml #apple #bbl **#thestruggle**
i have the cutest nephew in the world !url	#nephew **#cute** #family	#socute **#cute** #puppy

Table 1: Examples of top predictions from the models. The correct hashtag(s) if detected are in bold.

	word	*tweet2vec*
d_t, d_h	200	500
Total Parameters	3.91M	3.90M
Training Time / Epoch	**1528s**	9649s

Table 2: Model sizes and training time/epoch

4.3 Implementation Details

Word vectors and character vectors are both set to size $d_L = 150$ for their respective models. There were 2829 unique characters in the training set and we model each of these independently in a character look-up table. Embedding sizes were chosen such that each model had roughly the same number of parameters (Table 2). Training is performed using mini-batch gradient descent with Nesterov's momentum. We use a batch size $B = 64$, initial learning rate $\eta_0 = 0.01$ and momentum parameter $\mu_0 = 0.9$. L2-regularization with $\lambda = 0.001$ was applied to all models. Initial weights were drawn from 0-mean gaussians with $\sigma = 0.1$ and initial biases were set to 0. The hyperparameters were tuned one at a time keeping others fixed, and values with the lowest validation cost were chosen. The resultant combination was used to train the models until performance on validation set stopped increasing. During training, the learning rate is halved everytime the validation set precision increases by less than 0.01 % from one epoch to the next. The models converge in about 20 epochs. Code for training both the models is publicly available on github.

4.4 Results

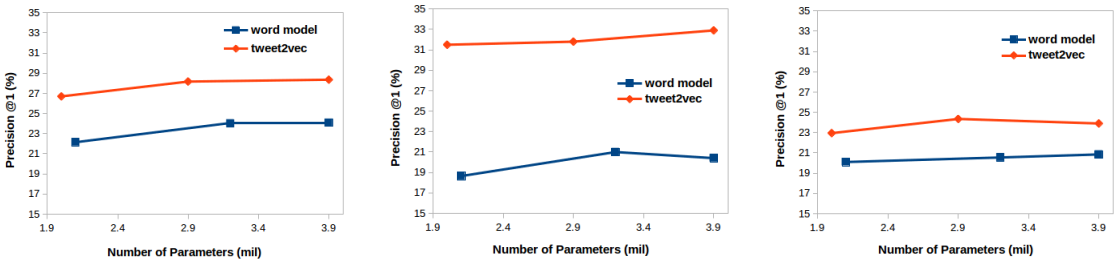
We test the character and word-level variants by predicting hashtags for a held-out test set of posts. Since there may be more than one correct hashtag per post, we generate a ranked list of tags for each post from the output posteriors, and report average precision@1, recall@10 and mean rank of the correct hashtags. These are listed in Table 3.

Model	Precision @1	Recall @10	Mean Rank
Full test set (50K)			
word	24.1%	42.8%	133
tweet2vec	**28.4%**	**48.5%**	**104**
Rare words test set (2K)			
word	20.4%	37.2%	167
tweet2vec	**32.9%**	**51.3%**	**104**
Frequent words test set (2K)			
word	20.9%	41.3%	133
tweet2vec	**23.9%**	**44.2%**	**112**

Table 3: Hashtag prediction results. Best numbers for each test set are in bold.

To see the performance of each model on posts containing rare words (RW) and frequent words (FW) we selected two test sets each containing 2,000 posts. We populated these sets with posts which had the maximum and minimum number of out-of-vocabulary words respectively, where vocabulary is defined by the 20K most frequent words. Overall, *tweet2vec* outperforms the word model, doing significantly better on RW test set and comparably on FW set. This improved performance comes at the cost of increased training time (see Table 2), since moving from words to characters results in longer input sequences to the GRU.

We also study the effect of model size on the performance of these models. For the word model we set vocabulary size V to 8K, 15K and 20K respectively. For *tweet2vec* we set the GRU hidden state size to 300, 400 and 500 respectively. Figure 2 shows precision 1 of the two models as the number of parameters is increased, for each test



(a) Full Test Set (b) Rare Words Test Set (c) Frequent Words Test Set

Figure 2: Precision @1 v Number of model parameters for word model and *tweet2vec*.

Dataset	# Hashtags	word	*tweet2vec*
small	933	28.0%	**33.1**%
medium	2039	24.1%	**28.4**%
large	5114	20.1%	**24.6**%

Table 4: Precision @1 as training data size and number of output labels is increased. Note that the test set is different for each setting.

set described above. There is not much variation in the performance, and moreover *tweet2vec* always outperforms the word based model for the same number of parameters.

Table 4 compares the models as complexity of the task is increased. We created 3 datasets (small, medium and large) with an increasing number of hashtags to be predicted. This was done by varying the lower threshold of the minimum number of tags per post for it to be included in the dataset. Once again we observe that *tweet2vec* outperforms its word-based counterpart for each of the three settings.

Finally, table 1 shows some predictions from the word level model and *tweet2vec*. We selected these to highlight some strengths of the character based approach - it is robust to word segmentation errors and spelling mistakes, effectively interprets emojis and other special characters to make predictions, and also performs comparably to the word-based approach for in-vocabulary tokens.

5 Conclusion

We have presented *tweet2vec* - a character level encoder for social media posts trained using supervision from associated *hashtags*. Our result shows that *tweet2vec* outperforms the word based approach, doing significantly better when the input post contains many rare words. We have focused only on English language posts, but the character model requires no language specific preprocessing and can be extended to other languages. For future work, one natural extension would be to use a character-level decoder for predicting the hashtags. This will allow generation of hashtags not seen in the training dataset. Also, it will be interesting to see how our tweet2vec embeddings can be used in domains where there is a need for semantic understanding of social media, such as tracking infectious diseases (Signorini et al., 2011). Hence, we provide an off-the-shelf encoder trained on *medium* dataset described above to compute vector-space representations of tweets along with our code on github.

Acknowledgments

We would like to thank Alex Smola, Yun Fu, Hsiao-Yu Fish Tung, Ruslan Salakhutdinov, and Barnabas Poczos for useful discussions. We would also like to thank Juergen Pfeffer for providing access to the Twitter data, and the reviewers for their comments.

References

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *The Journal of Machine Learning Research*, 3:1137–1155.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

Fréderic Godin, Viktor Slavkovikj, Wesley De Neve, Benjamin Schrauwen, and Rik Van de Walle. 2013. Using topic models for twitter hashtag recommendation. In *Proceedings of the 22nd international conference on World Wide Web companion*, pages 593–596. International World Wide Web Conferences Steering Committee.

Mihajlo Grbovic, Nemanja Djuric, Vladan Radosavljevic, Fabrizio Silvestri, and Narayan Bhamidipati. 2015. Context-and content-aware embeddings for query rewriting in sponsored search. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 383–392. ACM.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Andrej Karpathy, Justin Johnson, and Fei-Fei Li. 2015. Visualizing and understanding recurrent networks. *arXiv preprint arXiv:1506.02078*.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. 2015. Character-aware neural language models. *arXiv preprint arXiv:1508.06615*.

Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S Zemel, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. 2015. Skip-thought vectors. *arXiv preprint arXiv:1506.06726*.

Quoc V Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. *arXiv preprint arXiv:1405.4053*.

Wang Ling, Tiago Luís, Luís Marujo, Ramón Fernandez Astudillo, Silvio Amir, Chris Dyer, Alan W Black, and Isabel Trancoso. 2015. Finding function in form: Compositional character models for open vocabulary word representation. *arXiv preprint arXiv:1508.02096*.

Thang Luong, Richard Socher, and Christopher D Manning. 2013. Better word representations with recursive neural networks for morphology. In *CoNLL*, pages 104–113. Citeseer.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Li-Qiang Niu and Xin-Yu Dai. 2015. Topic2vec: Learning distributed representations of topics. *arXiv preprint arXiv:1506.08422*.

Cicero Nogueira dos Santos and Victor Guimarães. 2015. Boosting named entity recognition with neural character embeddings. *arXiv preprint arXiv:1505.05008*.

Cicero D Santos and Bianca Zadrozny. 2014. Learning character-level representations for part-of-speech tagging. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1818–1826.

Alessio Signorini, Alberto Maria Segre, and Philip M Polgreen. 2011. The use of twitter to track levels of disease activity and public concern in the us during the influenza a h1n1 pandemic. *PloS one*, 6(5):e19467.

Jason Weston, Sumit Chopra, and Keith Adams. 2014. tagspace: Semantic embeddings from hashtags. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1822–1827.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems*, pages 649–657.