Domain-Specific Tagging and FastText to Address Out of Vocabulary Words

Hugo Scurti
Computer Science
McGill University
hugo.scurti@mail.mcgill.ca

Isaac Sultan
Computer Science
McGill University
isaac.sultan@mail.mcgill.ca

Abstract

A methodology is presented to address the issue of out-of-vocabulary words for a LM task. The dataset is tagged during preprocessing, with one annotation specific to file system paths and the other specific to URLs. Chen et al.'s Enhanced LSTM method is trained on the Ubuntu Corpus v2.0, using a combination of pre-trained word embedding vectors and word embedding vectors trained on the corpus following an annotation process. The FastText model is used to produce the trained vectors, and was chosen for its use of character n-grams enrichment. The model produced a Mean Reciprocal Rank of 0.802, and is shown to produce increased performance on the corpus over a baseline with statistical significance (p = 2.2e-16).

1 Introduction

Language Modeling (LM) is the process of generating a probability distribution from a sequence of words. This field has great relevance to current applications within Natural Language Processing. LMs play a central role in speech recognition, a task that has much recent interest given its usage in mobile assistant applications, such as iOS 'Siri'. The presence of out-of-vocabulary (OOV) words is a serious problem for LM tasks. Since it is impossible to map every word in a domain to a meaning, various methodology have been developed to address the OOV issue.

The motivation of this paper is to test a methodology that could address such issues when modeling dialogue systems. The usage of novel 'special token tags' may grant improved performance to the ESIM model, a classifier built with biLSTM blocks (Chen et al., 2017). In addition to the end-of-utterance (eou) and end-of-turn (eot) tags , new tags were added during preprocessing step to denote the presence and content of word tokens, referring to file system paths and URLs. Three variations of the model are implemented, in order to demonstrate significance of increasing the richness of morphological features on performance.

The most rich model, v3, uses a combination of pre-trained and trained word embedding (WE) vectors. The trained WEs use the FastText model (Joulin et al., 2016), which enriches the embedding with character-level representations. Every model has its performance measured by 1 in 10 Recall@k for R@1, R@2 and R@5, as well as Mean Reciprocal Rank (MRR). We verify that harnessing domain knowledge and morphological information can address the OOV problem to ultimately improve the performance of the ESIM.

2 Related work

Chen et al. show that their ESIM model produces superior performance in Natural Language Inference (NLI) tasks in comparison with other models composed of LSTM units. Moreover, the ESIM is shown to outperform other recurrent memory blocks, such as GRUs (Gated Recurrent Units). The ESIM is described as a more simple model, yet still achieves state-of-the art performance in NLI. BiLSTMs (Schuster and Paliwal, 1997) are employed as one of the building blocks of the model. In the context representation layer, the BiLSTM encodes an input premise and hypothesis when the context and response embedding vector sequences are fed in. Here, the model learns to represent a word and its local sequence context. The attention matching layer then computes the similarity of hidden states between context and response to generate a co-attention matrix. Attended response vectors are generated for each word in context, i.e. the relevant semantics in the hypothesis is identified and composed for a hidden state of a word in a premise, with this operation repeated, too, for each word in a hypothesis. Local information is further enhanced concatenating the original vectors with vectors of the difference and element-wise product. The matching aggregation layer aggregates response-aware context and response representation with another BiLSTM. Max pooling is combined with the concatenation of forward and backward vector to form the final fixed vector v. Finally, v is fed into a fully-connected feedforward neural network (NN) with tanh activation functions within its hidden layer and softmax output layer to form a prediction. A multi-class cross-entropy loss is used to train the model.

In an ICLR 2018 submission paper (Anonymous, 2018), an enhanced version of the ESIM, named the ESIM^e is implemented for use for LM. Rather than randomly initializing OOV words with Gaussian samples, WEs are integrated, after being generated on the training corpus using Word2Vec (Mikolov et al., 2013) (d1 dimensions). Pre-trained WEs are sourced from GloVe (d2 dimensions) (Pennington et al., 2014). The vectors are then concatenated algorithmically to generate a dictionary of WEs with dimensionality d1 + d2. Character-composed embeddings are also integrated. The character-composed embedding is generated by concatenating the final state vector of the forward and backward direction of the BiLSTM.

The model is evaluated on the Ubuntu Dialogue Corpus v2.0 (UD2)(Lowe et al., 2015). The corpus was built in order to fulfill the following criteria: two-way, human-human conversation, large number of conversations (10^6), multi turn conversations (more than 3), with a task-specific domain, as opposed to chatbot systems. These properties make the corpus especially useful for the training of AI agents in targeted applications, and the size of dataset in particular allows NN architectures to be harnessed. The aforementioned ESIM^e when evaluated without the use of eou and eot, resulted in poorer performance, suggesting the importance of these domain-specific tags.

A similarity was noted between the linguistic challenges presented by UD2, and those of the micro-blogging platform 'Twitter'. Common errors are noted that have previously hindered the construction of a 'Twitter' Part-of-Speech (POS)(Derczynski et al., 2013). These include the proliferation of proper nouns, slang, tokenization error, genre-specific words and misspellings. Solutions suggested for some these errors are expanding the training set, and a normalization procedure for problematic tokens.

The FastText word representation enriches its WEs with subword information (Bojanowski et al., 2016). Unlike previous WE models which ignore the internal structure of words, FastText learns representations for character n-grams, and thus incorporates morphological information. The continuous skip-gram model (Witten et al., 2016) is extended so that each word is represented as a bag of character n-gram. All n-grams between 3 and 6 characters of length are extracted and a new scoring function represents a word by the sum of the vector representations of its n-grams. Word vectors for OOV words are generated by averaging vector representations of its n-grams. Bojanowski et al.'s paper verifies that this method computes non-trivial representations by computing the cosine similarity of each pair of n-grams that appear in word pairs in the English RW similarity dataset where one of the words are not in the training vocabulary.

The methodology that we present builds upon the ESIM, a simple yet extremely powerful model for the LM task. We maintain the use of UD2 as the training corpus, on account of its large size and the challenge it presents, given the texts' highly specialized, yet colloquial nature of conversation. Taking inspiration from Twitter POS tagging, we tag the dataset in an attempt to minimize two large sources of noise in the model. Rather than use Word2Vec, following the example of the ICLR 2018 submission, FastText was chosen, suitable not only as a word embedding vector trained on the corpus but also because it models the dataset at a character-level. Thus our methodology incorporates the features of ESIM^e's enhanced embeddings, without the difficulty of implementation and computational overheads.

3 Method

Data

UD2 was used to train and test the model. Source code provided by Lowe et. al was used to download and extract the given data samples[1]. The script `create_ubuntu_dataset.py` was called 3 times to generate the training, validation and test sets. The default parameters of the script were used, which included a random seed value of '1234' and no preprocessing.

The generated file 'train.csv' contains 1 million samples, with each sample containing a context sentence, a hypothesis utterance, and label of the relation between the context and the hypothesis: 1 if this response is the next utterance, 0 if it is a distractor. The training set has an equal distribution of labels. The validation and test set contain batched samples where each sample contains the context sentence, the true utterance, and 9 additional utterance which are distractors. Therefore, when formatting the data, each batched sample was split into 10 samples in the same format as the training set. This grants both the validation and test sets a 10% distribution of samples with label 1 and 90% distribution of samples with label 0. The expanded validation set contains 195,600 samples and the expanded test set contains 189,200 rows.

Preprocessing

The core of our hypothesis lies in the preprocessing step. The data are annotated with tags before tokenizing and lemmatizing samples. 3 datasets are generated; the first is unannotated, in order to train a baseline model, whilst the others include the addition of tags.

Regular expression (regex) is used to find all possible URLs and file system paths that respect a predefined format. In dataset 2, an annotation token is inserted following the token matched by the regex query. The tags are `__url__` for urls and `__path__` for paths. For the second version (advanced annotations), the tagset is extended so that for each URL, an extra tag is added which corresponds to the domain server name. For example, for the URL *www.google.com*, the tag would be `__google__`. The additional information the annotation provides will be reflected when constructing word embeddings trained on these annotated corpora.

After adding the annotations, standard tokenization and lemmatization procedures are applied. Given the colloquial nature of words in the dataset, the 'TweetTokenizer' found in the NLTK library (Bird et al., 2009) is used to separate the strings into tokens, setting the preserve_case parameter to false. Following tokenization, WordNetLemmatizer is used to lemmatize the tokens. At this stage, unique tokens are saved in a set, which is later used to filter unused word embeddings from the pretrained ones and to assign unique indices which are used to populate the embedding matrix.

Word embeddings

WE models were trained on the preprocessed training set to produce a vector of size 100. Both Word2Vec, and the aforementioned FastText model, were trained using the 'Gensim' toolkit . To minimize the size of the GloVe model dataset [2], unused tokens were removed. This resulted in a dataset of 430 MB, containing 9.08% of the WEs from the original GloVe dataset. We followed an algorithm proposed by the authors of an ICLR 2018 submission paper in order to concatenate the 300d pretrained GloVe vectors with the word embeddings trained on the UD2 corpus (Anonymous, 2018), resulting in 400d WEs with domain-specific knowledge.

Model

Williams et. al's (Williams et al., 2017) implementation of the ESIM model was adapted [3]. Since the code is licensed under the Apache license, a copy of this license is included in our source code. Our implementation was modified to be trained for binary classification, rather than the multi-class problem that the source code was implemented for. Changes

[1]https://github.com/rkadlec/ubuntu-ranking-dataset-creator

[2]Common Crawl (42B tokens, 1.9M vocab, uncased, 300d vectors, 1.75 GB download) https://github.com/stanfordnlp/GloVe

[3]Source code can be found here: https://github.com/nyu-mll/multiNLI

included altering the output of the last feed-forward layer to be of dimension 1 instead of 3, and using a sigmoid cross entropy function rather than a softmax cross entropy function.

Training

The model is implemented in TensorFlow (Abadi et al., 2015) (tensorflow-gpu version 1.4.1). ADAM optimization (Kingma and Ba, 2014) was used during training. The learning rate was set to 0.001. The batch size was 128. The number of hidden units of BiLSTM for word embeddings in both context representation and matching aggregation layers was 200. In the prediction layer, we used *ReLU* activation instead of *tanh* activation. In contrast with Williams et. al, we did not use dropout nor regularization, and we did not use early stopping. Finally, we used a maximum sequence length of 100 and 120 - a compromise between the value used by Williams et. al and the ICLR 2018 paper's authors, 50 and 180 characters respectively.

We implemented algorithms to evaluate R@k and MRR. The output of the ESIM was a one-dimensional vector with a probability for each each response. This was cast as a numpy array and then joined as a column of a data-frame, with the other columns as 'context', 'response' and 'label'. The pandas library (McKinney, 2011) was used as it allows fast sorting by column. The data-frame is first sorted by prediction and then by context. This now grants the most likely responses for each context in descending order. Then, by evaluating these 10 responses at a time, the desired metrics could be measured.

When evaluating these metrics, checkpoints of both the last trained model and the best model were stored. These models could be restored at any point, allowing training to be paused and parameters adjusted. Moreover, metrics at each number of batch steps were stored in a logging file so that training could be visualized. Training was halted given signs of overfitting and final results were predicted on a validation set.

In total, 4 variations on the model were tested:

Vers.	Pre-trained WE.	Trained WE.	Tags	Sequ.
v1	GloVe	Word2vec	N/A	100
v2	GloVe	Word2vec	Basic	100
v3	GloVe	FastText	Adv.	100
v3+	GloVe	FastText	Adv.	120

Table 1: Adv.: Advanced Tags, Sequ.: Max. sequence length

4 Results

A number of metrics were computed on the results of the validation set in order to evaluate the model: R@1, R@2, R@5, and MRR.

R@k is computed by searching the top 'k' responses (ranked by prediction score). A positive result occurs if the true response for a context is contained within this subset. The number of positive results is then divided by the number of contexts. MRR is computed as the reciprocal of the rank of the true response for each context, with the final metric displaying an average of these.

Table 2: Results for models after training completed

Model	R@1	R@2	R@5	MRR
v1	0.658	0.798	0.946	0.777
v2	0.673	0.806	0.949	0.787
v3	0.666	0.801	0.946	0.782
v3+	0.694	0.825	0.953	0.802

In addition, a subset of the validation set was generated, comprised only of the samples that contained URLs and/or paths, in order to evaluate the annotated models more closely. We evaluated again the same metrics on these filtered data. An increase in accuracy on these samples can be observed when tag annotations are used.

Table 3: Results on filtered validation set

Model	R@1	R@2	R@5	MRR
v1	0.652	0.791	0.941	0.772
v2	0.667	0.798	0.946	0.781
v3	0.662	0.797	0.943	0.778
v3+	0.692	0.822	0.952	0.801

As indicated by the learning curve in Fig. 1, the best model for v3+ is produced at around 22 000 batch steps. Beyond this point, both curves for Recall @ 1 and MRR begin to decline. The largest change in model performance is observed within the

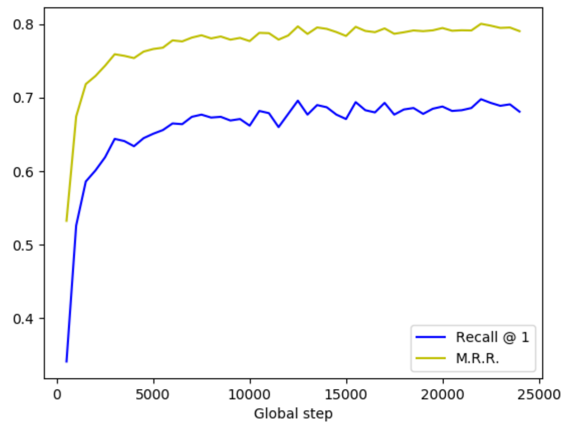

Figure 1: Recall@1 and M.R.R. calculated on the test set during training

first 10,000 batch steps, where backpropagation begins to converge and the majority of weights begin to converge. The R@1 metric creates a dichotomous outcome, either the highest ranked response for a given context is the true response, or else it is a distractor. 2X2 Contingency tables were constructed which tabulate the outcomes of two models on the same test set.

Table 4: McNemar Table on v3+ compared to v1

	False v3+	True v3+
False v1	4,560	2,130
True v1	1,420	11, 400

The McNemar's test(Hollander et al., 2013), a non-parametric test used on paired nominal data, was used to verify our hypothesis that the use of domain-specific tagging, and character enriched n-gram word embedding models, increase performance on this LM task.

$$H_0 : p_b = p_c \tag{1}$$

$$H_1 : p_b \neq p_c \tag{2}$$

Using the values from 4, the McNemar Test Statistic is:

$$\chi^2 = \frac{(c_{01} - c_{10})^2}{c_{01} - c_{10}} = 141.24 \tag{3}$$

Therefore, we reject the null hypothesis that assumes v1 and v3+ perform equally well (p = 2.2e-16 ¡ 0.05).

5 Discussion and conclusion

Thus, there is significant evidence to suggest that the use of domain-specific tagging, and character enriched n-gram WE models increase performance on LM tasks. Moreover, we expect our methodology to produce similar performance increases when applied to similar tasks on other corpora where the presence of OOV words is problematic.

When comparing the results that we received from v3+ to those produced by the ESIM^e using enhanced embeddings which had a MRR of 0.831, we note that our model does not present state-of-the-art performance. However, our methodology presents a key advantage in its simplicity of training of WE models with character n-grams. Unlike ESIM^e, no changes to the learning architecture were needed in order to train these features. Thus, the methodology may be equally applicable for improving performance on various other models, even if they possess vastly different learning architectures.

As demonstrated by the increased performance in v3+ in comparison with v3, the sequence length hyperparameter seems to be a large factor in training a powerful model. Contexts from UD2 can often reach a word length higher than 100. We were not able to implement the model with a sequence length of 180, due to RAM constraints. This may have had a significant impact on our results. It would, therefore, be worthwhile to investigate the relationship between model performance and this hyperparameter, and find an optimal sequence length that maximizes performance, with all other hyperparameters fixed. Thus, a sound heuristic could be developed to determine a sequence length value, given the properties of a dataset in question.

6 Statement of contributions

Hugo Scurti's main responsibilities were the generation and preprocessing of the corpus, the adaptation of the ESIM using TensorFlow and the writing of the methodology section of the report, and generating visualizations.

Isaac Sultan's main responsibilities were synthesizing and sharing research, implementation of the evaluation metrics, performing statistical analysis and writing majority of the report.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Anonymous. 2018. Enhance word representation for out-of-vocabulary on ubuntu dialogue corpus. *International Conference on Learning Representations*.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural Language Processing with Python*. O'Reilly Media, Inc., 1st edition.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. *CoRR*, abs/1607.04606.

Qian Chen, Xiaodan Zhu, Zhenhua Ling, Si Wei, Hui Jiang, and Diana Inkpen. 2017. Enhanced lstm for natural language inference. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL 2017)*, Vancouver, July. ACL.

Leon Derczynski, Alan Ritter, Sam Clark, and Kalina Bontcheva. 2013. Twitter part-of-speech tagging for all: Overcoming sparse and noisy data. In *RANLP*.

Myles Hollander, Douglas A Wolfe, and Eric Chicken. 2013. *Nonparametric statistical methods*. John Wiley & Sons.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Ryan Lowe, Nissan Pow, Iulian Serban, and Joelle Pineau. 2015. The ubuntu dialogue corpus: A large dataset for research in unstructured multi-turn dialogue systems. *arXiv preprint arXiv:1506.08909*.

Wes McKinney. 2011. pandas: a foundational python library for data analysis and statistics.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681.

Adina Williams, Nikita Nangia, and Samuel R Bowman. 2017. A broad-coverage challenge corpus for sentence understanding through inference. *arXiv preprint arXiv:1704.05426*.

Ian H Witten, Eibe Frank, Mark A Hall, and Christopher J Pal. 2016. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann.