Best of both worlds: Combining distributional and definitional word vectors

Anonymous EMNLP submission

Abstract

Current popular word representations are *distributional*, meaning that they are derived from co-occurrence statistics. While successful, the distributional approach has limitations – it requires a large training corpus, and provides no representation for out-of-vocabulary words at test time. By contrast, *definitional* word representations – i.e. representations derived from dictionary definitions of words – require less training data and can be computed on-the-fly. We use a neural auto-encoder model to obtain definitional word vectors, and show that they capture complementary information to distributional word vectors. We demonstrate experimentally that a combination of distributional and definitional word vectors provide an improvement for Neural Machine Translation.

1 Introduction

Pre-trained word representations that capture distributional semantics have contributed enormously toward advances in natural language processing (Mikolov et al., 2013) (Pennington et al., 2014). However, there are a number of limitations. Perhaps most significantly, these word vectors are unable to handle out-of-vocabulary (OOV) words – that is, rare or jargon words not built into the pre-trained list of vectors. Additionally, there are unintuitive properties of the vector spaces captured by distributional semantics (for example, words that are antonyms often end up having very similar representations).

Meanwhile, alternative non-distributional approaches to word representation have also been proposed (Faruqui and Dyer, 2015). A particularly intuitive non-distributional representation is the *definitional* word representation – that is, conveying the word of the meaning with a sentence that directly states what the word means. Since both definitions and word vectors attempt to convey the semantic meaning of a given word, it makes intuitive sense that it should be possible to generate vectors directly from definitions. Surprisingly, little work has been done on leveraging word definitions for general-purpose word vectors. While attempts at definitional word vectors have shown promise in capturing semantics, the marginal benefit of including them has not been adequately explored (Bahdanau et al., 2017) (Hill et al., 2015).

We build upon their work with an autoencoder model to capture word semantics from definitions, and additional evaluation on the popular downstream task of neural machine translation (NMT). We show that definitional vectors alone are unable to perform as well as distributional vectors. This motivates us to propose our main contribution, which is to combine distributional and definitonal word vectors. Including both types of representation captures different aspects of a given word's meaning and so the combination of them may outperform either one alone. Moreover, we have flexibility in dealing with out-of-vocabulary words by using constructing definitional vectors on-the-fly. We support this hypothesis with intrinsic and extrinsic evaluation. We present this as a work in progress, with planned future work is the natural extension of beyond dictionaries as sources of input and into spellings, Wikipedia articles, etc.TODO

2 Related Work

There have been a number of prior works toward deriving word vectors from dictionary definitions. The closest to our work is Bahdanau et al. (2017), in which the authors leverage dictionary definitions and character-level morphology to construct neural models that can embed word vectors on-the-fly. However, their approach was limited by

the fact their definition encoding was based on training for only one extrinsic task, which intuitively may result in task-specific vectors that do not generically capture the meaning of the word. Our model differs from theirs in our use of an auto-encoder for embedding definitions, The authors also briefly explore combinations of definitional and distributional word vectors, but did not focus nor analyze it at length. We go further in motivating the idea behind combining word vectors, showing performance on both intrinsic and extrinsic tasks, and analyzing some qualitative differences between the three types of vectors.

Other related works that have explored derivation of non-distributional word vectors. Most salient is the neural model by Hill et al. (2015) which takes a similar approach of embedding definitions and shows success at the reverse lookup task, but evaluates performance on translation through a bilingual embedding instead of augmenting word vectors. Work by Tissier et al. (2017) leverages dictionary definitions, but implements a skip-gram model based on sampling "positive" and "negative" pairs instead of directly embedding definitions through a recurrent model. Definition-derived word embeddings were combined with language modeling in Noraset et al. (2016), in which the authors demonstrate success at modeling the definition of a word given its embedding. Other attempts to use semantic knowledge for word embeddings include Xu et al. (2014), Zhou et al. (2015), Rothe and Schütze (2015), and Faruqui and Dyer (2015) .

More broadly, there have been several recent attempts to address the out-of-vocabulary problem. Notably, character-level neural language models (Kim et al., 2015) have seen much success. For machine translation specifically, a recently proposed approach for handling out-of-vocabulary words is to combine a word-level model with a character-level model (Luong and Manning, 2016). While character-level models are certainly effective at capturing morphological meaning, but we hypothesize that definition-based word embeddings can capture more nuance for semantically rich words.

3 Methods

3.1 Definition encoder

Our main definition encoding model is implemented as a Seq2Seq auto-encoder model with attention (Bahdanau et al., 2014). Given an input word w, we look up its definition $d(w)$ in the database of definitions. Each word of the definition is encoded through an embedding layer (trained from scratch), and run through the Seq2Seq model to produce $f(w)$. We minimize negative log-likelihood between the predicted $f(w)$ and the ground truth definition $d(w)$ to train the auto-encoder model. The last hidden state of the Seq2Seq model is employed as our predicted definitional word embedding.

In addition to the auto-encoding model, we train a recurrent LSTM model to maximize the bag-of-words unigram probability for each word d in the definition $d(w)$ (via minimizing negative log-likelihood for the words in the definition)$similartothebaselineworkin(BoscandVincent).Weencodermodel, andnotethattheauto-encodermodelhasthefreedomtodefineitsownlatentspace, no$

3.2 Neural Machine Translation

Our approach for machine translation is another Seq2Seq model with attention, implemented through Harvard's open-source OpenNMT project (Klein et al., 2017). We use the default plain RNN encoder and decoder with attention and LSTM cells. To leverage our dictionary-derived definitions, we concatenate GloVe vectors $g(w)$ and our embedded vectors $f(w)$ together when training and evaluating the model.

4 Experiments

4.1 Data

For definitions, we follow the practice of previous work and employ data from the WordNet database (Miller, 1995). For the LSTM baseline model, our ground-truth embeddings are the corresponding 100-dimensional GloVe vectors (Pennington et al., 2014); we use the 400k vocabulary version trained on Wikimedia 2014 and Gigaword 6. Lastly, for the NMT task we make use of the Yandex 1M English-Russian Corpus which has one million aligned English and Russian sentences (Yandex, 2018).

4.2 Training

Our Seq2Seq model employs a hidden state of [Todo].

Our LSTM employs a hidden state of dimension 150 and has 2 layers, with a dropout probability of 0.3 between each layer. Our word embedding

layer is initialized from 100-dimensional GloVe vectors and fine-tuned. Unknown words that occur in the definitions are represented by <UNK> tokens and an embedding for this token is trained as well. We implemented our model in PyTorch (Paszke et al., 2017) and trained using the Adam (Kingma and Ba, 2014) optimizer for 15 epochs with a learning rate of 0.0001 and a batch size of 64.

The full dataset that we train upon is the set of 400K pre-trained GloVe words

4.3 Intrinsic evaluation

We evaluated on the L_2 loss in our test set constructed from the Stanford Rare Words dataset and performed an ablation analysis of architectural choices we made (Table 1). The ablation experiments compare performance of the "standard" model we have described thus far against versions with (1) bidirection removed, (2) attention removed, and (3) the GRU layer removed (limiting the model to attention only). Each experiment was run with the same set of hyper-parameters described.

We note that the average distance between any two random vectors in GloVe is 2.66, and a baseline approach of straightforwardly averaging the word embeddings comprising a given definition achieves an average distance of 4.00. All neural models we assessed yielded significantly better performance, with our "standard" model outperforming all others at a loss of 0.180. Surprisingly, the version of our model without a GRU component was able to perform almost as well as the full model. In fact, ablating the attention mechanism resulted in comparatively worse performance than ablating the GRU component, suggesting that attention is the most critical choice we made in model architecture.

Qualitatively, we verify through t-SNE visualizations (van der Maaten and Hinton, 2008) of the test set embedding space that our predicted Def2Vec embeddings are able to accurately reconstruct the corresponding ground-truth GloVe vectors for most words (Figure **??**). Moreover, when we query Def2Vec with given definitions, the ground-truth vector is frequently among the nearest GloVe vectors in the embedded space (Figure **??**). Lastly, we confirm that the attention mechanism is intuitively reasonable, as it weighs the most relevant words of a given definition most significantly.

4.4 Extrinsic Evaluation

TODO We do extrinsic evaluation to verify how useful definitional vectors are for downstream tasks. We train our translation model derived from OpenNMT with 2 layers, a hidden state of size 200, and frozen encoder vectors. We run two sets of experiments: (1) a set with only a subset of the most frequent GloVe vectors, and (2) a set with a combination of the most frequent GloVe vectors and our vectors embedded by Def2Vec. We implemented our model in PyTorch (Paszke et al., 2017) and trained using the Adam (Kingma and Ba, 2014) optimizer for 12 epochs with a learning rate of 0.0001 and a batch size of 64.

TODO Quantitative results are presented in Table 2. We find that if we train the NMT model with only 1k of the most frequent words in GloVe, performance is noticably inferior. However, if we augment these words with the vectors imputed by Def2Vec, performance improves considerably – in fact, it is almost on par with perplexities reported using the top 10k most frequent words in GloVe. We make the same observation when we train the NMT model with 10k of the most frequent words in GloVe – the addition of Word2Vec embeddings improves the BLEU score. However, when we reach a vocabulary of 50k GloVe vectors as a baseline, the addition of our Def2Vec embeddings actually causes a slight decrease in performance of the model. This indicates that there are diminishing marginal returns to the addition of Def2Vec embeddings into the NMT system; it is most helpful when the original vocabulary is small and as we look up definitions for rarer words, the overall boost in performance is less significant.

We also present qualitative results in (Figure **??**). Qualitatively, our model is able to generate reasonably accurate translations much of the time. We found it especially surprising given that we had a hidden size of only 200 with little tuning, and trained on a relatively small dataset.

5 Conclusion

Motivated as an intuitive approach for ... Our experiments show that the approach is intrinsically successful at constructing word vectors, and can be used as a reverse lookup tool. Moreover, we verify that the resulting word embeddings

Ablation	**Standard**	No bidirection	No GRU	No attention	Baseline-GloVe	Baseline-Avg
L_2 Distance	**0.180**	0.184	0.184	0.186	2.66	4.00

Table 1: Mean distances for Def2Vec and GloVe vectors for words in Stanford Rare Words Dataset (Luong et al., 2013).

Vocabulary	400k	50k + Def2Vec	10k + Def2Vec	1k + Def2Vec	50k	10k	1k
Perplexities	38.15	39.19	39.27	45.23	39.05	41.82	59.64
BLEU	5.55	5.09	4.61	–	5.15	4.44	2.82

Table 2: Quantitative evaluation of NMT models evaluated on test set sentences. The vocabulary labels indicate whether only a subset of the most common GloVe vectors were used, or whether a combination of that subset was combined with Def2Vec vectors. The missing entry is due to the fact that we ran out of time to generate the last 1k + Def2Vec BLEU score.

have utility in the extrinsic neural machine translation task, especially when the original vocabulary is relatively small. We believe that dictionary-derived word embeddings can be especially useful for highly technical words, slang, and other languages – any instance in which a rich dataset of definitions is available.

There are a few important limitations in our work and possible extensions toward further work. First, we currently handle out-of-vocabulary tokens in given definitions with an <UNK> token – an inelegant approach for which we'd like to explore a recursive approach in the future, looking up dictionary definitions for unknown entries in a definition. Second, we currently include definitions in our training process that may reference semantically identical words to the target – for example, "transparency: the condition of being transparent". While we decided that it was reasonable to keep these definitions in for now, we would want to explore the impact of removing such definitions in the future. Finally, we are interested in exploring datasets beyond definitions, character-level words, including Wikipedia articles, parse trees, and dependency trees.

References

Dzmitry Bahdanau, Tom Bosc, Stanisaw Jastrzbski, Edward Grefenstette, Pascal Vincent, and Yoshua Bengio. 2017. Learning to Compute Word Embeddings On the Fly. *arXiv:1706.00286 [cs]*. ArXiv: 1706.00286.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

Tom Bosc and Pascal Vincent. Learning word embeddings from dictionary denitions only. page 6.

Manaal Faruqui and Chris Dyer. 2015. Non-distributional Word Vector Representations. *arXiv:1506.05230 [cs]*. ArXiv: 1506.05230.

Felix Hill, Kyunghyun Cho, Anna Korhonen, and Yoshua Bengio. 2015. Learning to understand phrases by embedding the dictionary. *CoRR*, abs/1504.00548.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M. Rush. 2015. Character-aware neural language models. *CoRR*, abs/1508.06615.

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

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M. Rush. 2017. Opennmt: Open-source toolkit for neural machine translation. In *Proc. ACL*.

Minh-Thang Luong and Christopher D. Manning. 2016. Achieving open vocabulary neural machine translation with hybrid word-character models. *CoRR*, abs/1604.00788.

Minh-Thang Luong, Richard Socher, and Christopher D. Manning. 2013. Better word representations with recursive neural networks for morphology. In *CoNLL*, Sofia, Bulgaria.

L.J.P. van der Maaten and G.E. Hinton. 2008. Visualizing high-dimensional data using t-sne.

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

George A. Miller. 1995. Wordnet: A lexical database for english. *Commun. ACM*, 38(11):39–41.

Thanapon Noraset, Chen Liang, Larry Birnbaum, and Doug Downey. 2016. Definition modeling: Learning to define word embeddings in natural language. *CoRR*, abs/1612.00394.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in pytorch.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Sascha Rothe and Hinrich Schütze. 2015. Autoextend: Extending word embeddings to embeddings for synsets and lexemes. *CoRR*, abs/1507.01127.

Julien Tissier, Christophe Gravier, and Amaury Habrard. 2017. Dict2vec : Learning Word Embeddings using Lexical Dictionaries. In *Conference on Empirical Methods in Natural Language Processing (EMNLP 2017)*, pages 254–263, Copenhague, Denmark.

Chang Xu, Yalong Bai, Jiang Bian, Bin Gao, Gang Wang, Xiaoguang Liu, and Tie-Yan Liu. 2014. Rc-net: A general framework for incorporating knowledge into word representations. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, CIKM '14, pages 1219–1228, New York, NY, USA. ACM.

Yandex. 2018. Yandex 1m en-ru dataset.

Chunting Zhou, Chonglin Sun, Zhiyuan Liu, and Francis C. M. Lau. 2015. Category enhanced word embedding. *CoRR*, abs/1511.08629.