Hybrid Distributional and Definitional Word Vectors

|  |  |
| --- | --- |
| Haiyuan Mei xxxx@stanford.edu | Ranjani Venkatesh Iyer xxx@stanford.edu |

Abstract

Word vectors are typically computed by implementing distributional statistics (such as co-occurrence), but it is surprising that the most logical source of words' meanings - dictionaries - are not leveraged in the process. We want to investigate the ability to integrate word definitions with distributional statistics in the process of creating word vectors. We first iterate and improve a seq2seq auto-encoder model that can act as a baseline method to obtain definitional word vectors and show that they capture complementary information to distributional word vectors, and then attempt to implement a variational autoencoder. We attempt to show that a combination of distributional and definitional word vectors produced from an autoencoder provide an improvement for Neural Machine Translation.

# Introduction

*(****TODO: general description needs to be updated****) 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. These word vectors are unable to handle out-of-vocabulary (OOV) words – that is, rare or jargon words not built into the pretrained list of vectors. Additionally, there are un-intuitive 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).*

This project is a continuation of Andrey Kurenkov and Tony Duan’s previous work, Def2Vec[ADD REFERENCE], in which the authors quantitatively and qualitatively demonstrated that leveraging definitions alone can be used to embed words into a semantically meaningful space comparable to GloVe embeddings; they also demonstrated the utility of Def2Vec in improving the performance of a Neural Machine Translation model when the pre-trained vectors vocabulary is limited and there are several out-of-vocabulary words.

However, since the definitions were just being used to recreate GloVe vectors, the model was of limited value since GloVe is already computed for a large portion of the english vocabulary. We realized that an intriguing direction to explore is creating an entirely new form of word vector that does not rely on distributional statistics and is instead only based on definitions - a ‘definitional’ word vector’. Intuitively, this should be doable by autoencoding the definitions, since a word’s definition encodes its meaning and so the latent vector of the encoded definition should be a valid word vector. It may be that these vectors contain information that that distributional vectors do not capture, which motivates the introduction of a combined distributional and definitional word vectors - Hybrid Distributional and Definitional Word Vectors. Including both types of representation can capture complementary aspects of a given word’s meaning, so the combined vector may outperform either one alone.

We first iterate and improve a seq2seq auto-encoder model that can act as a baseline method to obtain definitional word vectors and show that they capture complementary information to distributional word vectors, and then attempt to implement a variational autoencoder. We attempt to show that a combination of distributional and definitional word vectors produced from an autoencoder provide an improvement for Neural Machine Translation.

# Related Work

*[****TODO: Related work need to be updated****]There have been a number of prior works toward deriving word vectors from dictionary definitions. One such 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 Schutze¨ (2015), and Faruqui and Dyer (2015) .*

# Methods

As a continuation of a previous project, our plan is to extend the work that has already been done on creating ‘definitional’ word vectors. We will first iterate and improve a Seq2Seq autoencoder implementation that can act as a baseline method, and then attempt to implement a variational autoencoder.[[1]](#footnote-1) Lastly, we will evaluate various possible ways of combining the two types of vectors. If the combined vectors do well on intrinsic metrics, we will also attempt to evaluate them on extrinsic metrics via downstream tasks such as NMT. Different from Def2Vec which uses regression methods to calculate definitional vectors, we use the general autoencoding framework to learn definitional embeddings that represent the definitions of the associated word. The baseline seq2seq model is a continuation of the current work [ADD REFERENCE]; The more advanced Variational Autoencoder is based off of [ADD REFERENCE Bowman etc], which is an rnn-based variational autoencoder generative model that incorporates distributed latent representations of entire sentences. This factorization allows it to explicitly model holistic properties of sentences such as style, topic, and high-level syntactic features.

3.1 Seq2seq Autoencoder

[**TODO**: structure, loss function, implementation, etc] Our baseline autoencoder takes the form of a Seq2seq autoencoder (SAE) model that respects the initial syntactic structure of the sentence. Given an input word *w*, we look up its definition *d*(*w*). Each word of the definition is encoded through an embedding layer (trained from scratch) and then ran through a 2layer LSTM encoder to produce the dense representation *h* that represents the definitional embedding. The Seq2seq autoencoder model minimizes the negative log-likelihood between the predicted definitional word and the ground truth definitional word *d* for *every* position in the definition, thereby constraining the definitional embedding to also learn the relative syntactic placement/relationships of the words in the definition as shown in Figure 4.

3.2 Variational Autoencoder

Our second, more complex autoencoder is an extension of the RNNLM that is designed to explicitly capture global features of a sentence in a continuous latent variable. (**TODO**: structure, loss function, implementation, etc; Describe how sentence vectors are calculated and how such vector may outperform Seq2seq autoencoder)

3.4 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.

# Experiments

4.1 Data

For definitions, we follow the practice of previous work and employ data from the WordNet database (Miller, 1995). We use the 400k vocabulary version of GloVe trained on Wikimedia 2014 and Gigaword 6 (Pennington et al., 2014). These 400k words were used as the input words from which we used WordNet to generate definitions. Then the definitions were run through the two autoencoder models where the hidden state between the encoder and the decoder was used to represent the input GloVe word. 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 Autoencoder model employs a hidden state of size 200. [Add VAE part] For each model the encoder consists of two layers with a dropout probability of 0.3 at training time. The input word embeddings for each dictionary word were initialized randomly at the start of training and were updated during training. 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.

4.3 Intrinsic evaluation

**Similarity[[2]](#footnote-2) and Relatedness[[3]](#footnote-3)**: We evaluate the quality of the embeddings produced from our autoencoder models using relatedness and similarity metrics as is standard for word embeddings on a set of existing benchmarks. These benchmarks are evaluated on similarity and/or relatedness datasets that contain pairs of words and human annotated scores for each pair of words. The metric calculated in order to measure similarity and relatedness is the Spearman’s rank-order correlation, *rs*, which ranges between −1 and +1 (Table 1). An *rs* of +1 indicates high correlation and means that the embeddings are similar and related, and a *rs* of −1 implies poor performance on similarity and relatedness tasks.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Model Cosine Similarity  GloVe 0.64   |  |  | | --- | --- | | SAE | 0.65 | | VAE |  |   Table 2: Average cosine similarity for GloVe, SAE, and VAE embeddings over all antonym pairs. |

**Antonymy**: [Do we need it?]

**Qualitative**: [Do we need it?]

4.4 Extrinsic Evaluation

The purpose of extrinsic evaluations is to verify how useful definitional vectors are for downstream tasks. We train a translation model from English to Russian derived from OpenNMT with 2 layers, a hidden state of size 200, and frozen encoder vectors. We run three sets of experiments: (1) with GloVe vectors as embeddings, (2) with GloVe vectors concatenated with the BOW embeddings, and (3) with GloVe vectors concatenated with the SAE embeddings. 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. Quantitative results are presented in Table 3. The two metrics we measured the results of the NMT task are accuracy and perplexity. Accuracy is the percentage of tokens accurately translated by the model and perplexity is a measure of the prediction error in the translated tokens.

# Discussion

5.1 Intrinsic Evaluation

5.2 Extrinsic Evaluations

# Conclusion

# References

1. **Generating Sentences from a Continuous Space** [Samuel R. Bowman](https://arxiv.org/search/cs?searchtype=author&query=Bowman%2C+S+R), [Luke Vilnis](https://arxiv.org/search/cs?searchtype=author&query=Vilnis%2C+L), [Oriol Vinyals](https://arxiv.org/search/cs?searchtype=author&query=Vinyals%2C+O), [Andrew M. Dai](https://arxiv.org/search/cs?searchtype=author&query=Dai%2C+A+M), [Rafal Jozefowicz](https://arxiv.org/search/cs?searchtype=author&query=Jozefowicz%2C+R), [Samy Bengio](https://arxiv.org/search/cs?searchtype=author&query=Bengio%2C+S) [↑](#footnote-ref-1)
2. SimLex999 [[Hill et al.](http://webcache.googleusercontent.com/search?q=cache:BjB-WvggUYMJ:metalearning.ml/papers/metalearn17_bosc.pdf+&cd=1&hl=en&ct=clnk&gl=us&client=ubuntu#5), 2016] and SimLex333 [↑](#footnote-ref-2)
3. RG [[Rubenstein and Goodenough,](http://webcache.googleusercontent.com/search?q=cache:BjB-WvggUYMJ:metalearning.ml/papers/metalearn17_bosc.pdf+&cd=1&hl=en&ct=clnk&gl=us&client=ubuntu#6) 1965], WS353 [Finkelstein et al., 2001], SCWS Huang et al. [2012] and MTurk Radinsky et al., [[2011]](http://webcache.googleusercontent.com/search?q=cache:BjB-WvggUYMJ:metalearning.ml/papers/metalearn17_bosc.pdf+&cd=1&hl=en&ct=clnk&gl=us&client=ubuntu#6) Halawi et al. [2012]. [↑](#footnote-ref-3)