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.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) .*

# Approach

As a continuation of a previous project Def2Vec, we mainly focused on two of the existing models in creating definitional embeddings: an LSTM baseline model which is composed of a multi-layer LSTM encoder and a simple conditional language model decoder with each output trained by Cross-Entropy loss based on 1-hot-vector over the entire vocabulary and softmax output(can be seen as a simple classification problem); an normal Seq2seq Sentence Autoencoder model with both encoder and decoder a configurable recurrent neural network.

After obtaining different word embeddings separately, the intrinsic evaluation is done by a series of word embeddings benchmarks, comparison of LSTM baseline model, Seq2seq model and GloVe word embedding are done to give the evidence that the LSTM baseline model is roughly at the level of distributional method while the Seq2seq model shows limited evidence of such capability.

We finally applied our learnt word embeddings in combination with pretrained GloVe vectors to form our HybridVec embeddings, in the hope of capturing both distributional and definitional aspect of word vectors to improve downstream Natural Machine Translation systems. In our case we choose OpenNMT as our extrinsic evaluation system.

We also explored the possibility of utilizing a more advanced Variational Autoencoder model in creating definitional embeddings. 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.

# Models

4.1 LSTM baseline model

This LSTM baseline model contains two word embeddings separately for encoder and decoder each. Let be the set of all words that are used in definitions, and be the set of all words that are to be defined. and are not necessarily the same but will be from the same vocabulary set. The definition of each word w from is a list of words from denoted as =(,,...,) where is the index of a word in vocabulary . The definition for is a hence a sequence of words which is encoded by RNN with LSTM cells [Hochreiter and Schmidhuber, 1997]; multiple meanings will be encoded with multiple representations. The LSTM is parameterized by the input embedding which is a ||m matrix which the row an m-dimentional input embedding for the word of . Depends how many layers we pass to the model, the last hidden states will be the same number of m-dimensional definition embeddings as the layers. The model is depicted in Figure 1, and the hidden layer can be described as the following equation.

h = f E,θ (d) = LSTM E,θ (d)

The decoder part will have two types of inputs, the hidden state of encoder, and the sequence of output word embeddings corresponding to the word definitions. It is a simple conditional language model, with each of the next predicted word learned in the way of normal classification method, using softmax, |V^D| dimensional one-hot-vector and cross-entropy loss.

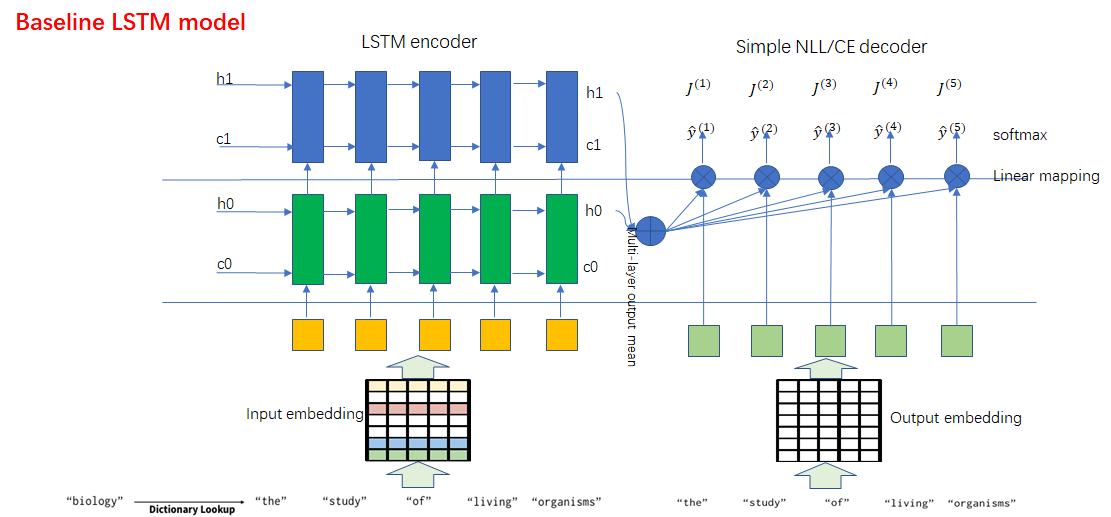
[================= ]

We made different input embeddings and output embeddings in this paper which is possible to cause overfitting problems; a unique word embedding matrix is an alternative way of implementation which is to be explored in future experiments.

The total loss of all word definitions is just the negative of sum over all sentences (can also be interpreted as a Negative Log-Likelihood Loss NLL.).

For the LSTM baseline model, in order to make our definition embeddings not far away from our learnt word embeddings, a penalty weighted by \lambda is applied on the euclidean distance between the predicted word embeddings and the learnt word embeddings, which gives the final loss function as:

[================= ]



4.2 Seq2seq Autoencoder

The second model we explored to create word embeddings 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 2 layer 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 definitions.

4.3 Variational Autoencoder (VAE)

We also explored the more fancy way of Variational Autoencoders (VAE). VAE 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)

4.4Neural 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 generate our HybridVec by concatenating GloVe vectors g(w) and our embedded vectors f(w) created by different methods when training and evaluating the model. The evaluation of the model is done by comparing NMT training results using pre-trained HybridVec and pre-trained GloVe embeddings.

# Experiments

5.1 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 word. Lastly, for the NMT task we make use of both the default 10k demo English-German OpenNMT corpus and the Yandex 1M English-Russian Corpus which has one million aligned English and Russian sentences (Yandex, 2018). The previous dataset is too small to make any serious NMT predictions but we believe it is a quick and dirty way of making comparison between our HybridVec embedding and GloVe embedding.

5.2 Training

We implemented our model in PyTorch (Paszke et al., 2017) and trained using the Adam (Kingma and Ba, 2014) optimizer for 20 epochs with a learning rate of 0*.*0001 and a batch size of 64. The full dataset that we train our definitional word embeddings upon is the set of 400K pre-trained GloVe words. We trained both LSTM baseline model and seq2seq model with 2 layer LSTM RNN and 300-dimensional embeddings and 150-dimensional hidden vectors.

5.3 Intrinsic evaluation

**Similarity[[1]](#footnote-1) and Relatedness[[2]](#footnote-2)**: We evaluate the quality of the embeddings produced from our autoencoder models by using a third party word embedding benchmark test toolsets: Word Embedding Benchmark (WEB). WEB is focused on evaluating and reporting results on common benchmarks (analogy, similarity and categorization). 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 predictions and the ground truth are ranked and the metric calculated in order to measure the ranks is Spearman’s \ro x 100.

Quantitatively the Word Embeddings Benchmarks for GloVe, LSTM Baseline and Seq2seq model reveals that our LSTM baseline model is roughly at the level of distributional method; while on the other hand the more complex Seq2seq SAE model shows very limited evidence of such capability.

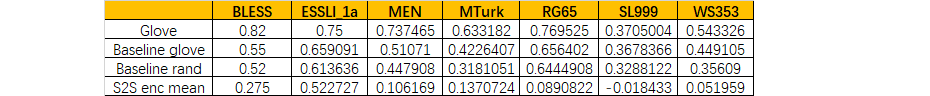
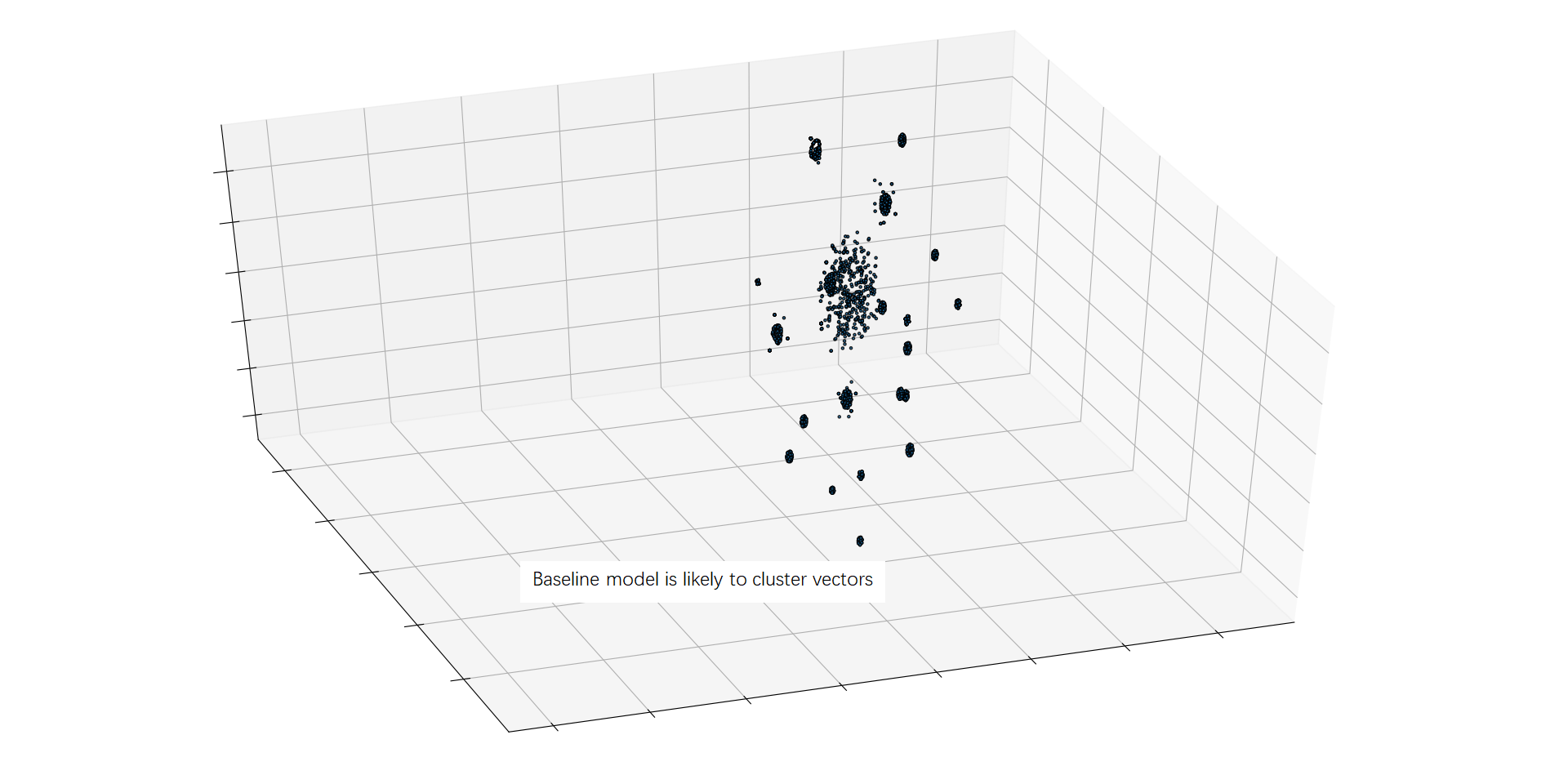
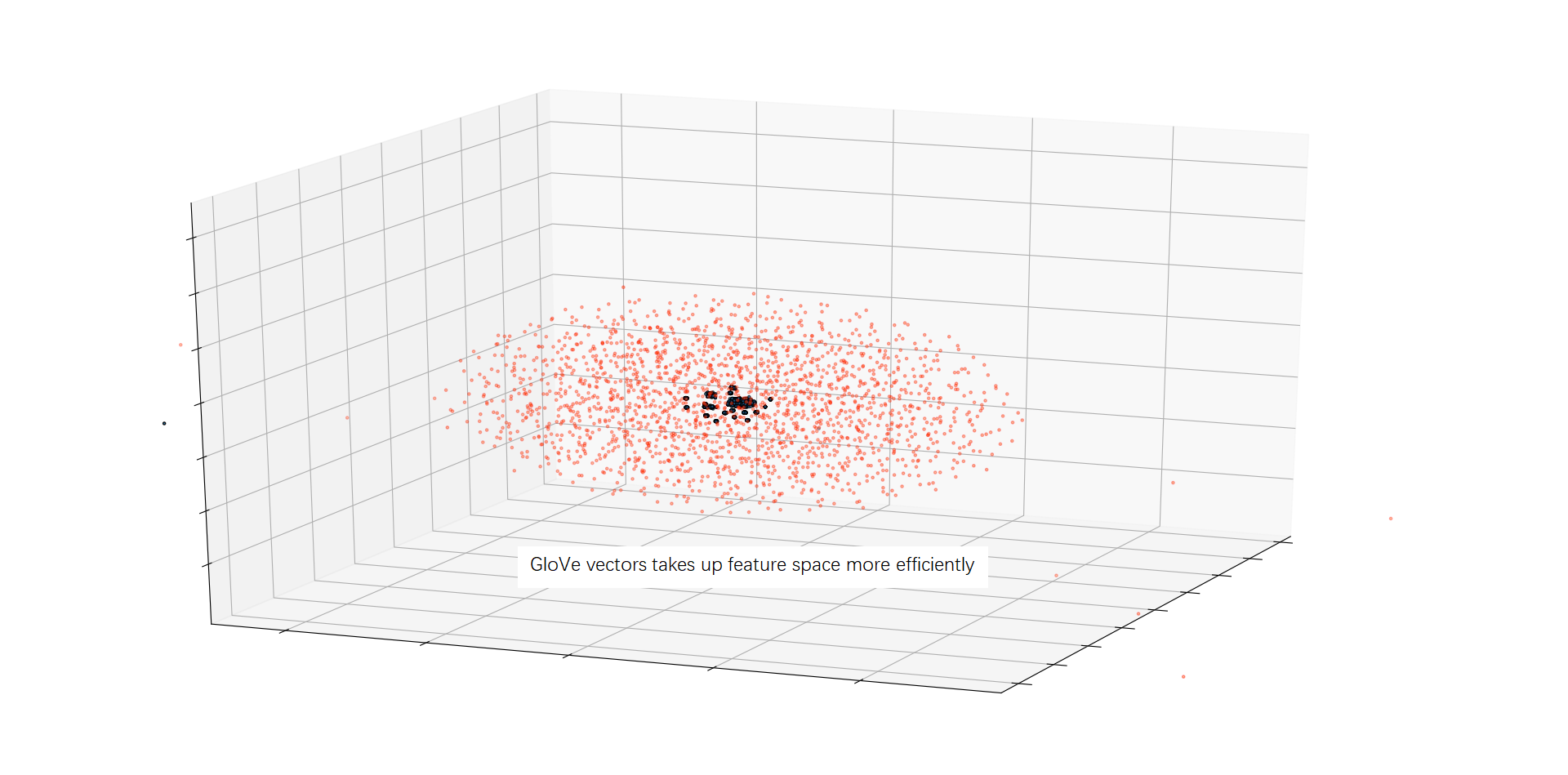


Table 1: Spearman’s ρ × 100 on various benchmarks. (GloVe: for GloVe vectors; Baseline glove: WEB benchmark for LSTM baseline model initialized from GloVe; Baseline rand: WEB benchmark for LSTM baseline model initialized randomly; s2s enc mean: WEB benchmark for Seq2seq model with encoder output mean as the def vec.)

Qualitatively, we explored through t-SNE visualizations (van der Maaten and Hinton, 2008) of the test set embedding space which reveals to us that our predicted LSTM baseline definitional word embeddings are likely to cluster in the feature space to a much smaller number (compared to vocabulary size) of clusters. It is probably because of the training definitions, compared to large corpus natural language datasets, are lack of variations in the sense of expressing much more complex and abundant human emotions, and probably will need to be trained from a broader text source. (Figure ??). The Glove embeddings on the other hand makes use of feature space more efficiently, which means GloVe vectors is capable of grasping more subtle meanings of words. 

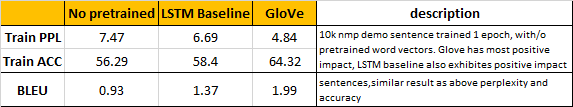
LSTM baseline vectors tend to cluster in feature space. Need to train from a broader source.



Glove makes use of feature space more efficiently, grasp more sutle meaning of words.

5.4 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. In order to explorer the potential improvement of HybridVec in downstream NMT task, we use the same GloVe 400K words as input vocabulary to generate embeddings for all the extrinsic evaluations. We first apply the 10K default demo English-German OpenNMT corpus to make a quick observation on separate pre-trained word embeddings, the quantitative results is shown in Figure[]



We finally come to the point of comparing our HybridVec word vectors with GloVe in NMT task. We apply the Yandex 1M English-Russian Corpus which has one million aligned English and Russian sentences (Yandex, 2018), to two pretrained embeddings for the NMT task: (1) with GloVe vectors as embeddings, (2) with GloVe vectors concatenated with the LSTM baseline embeddings. Quantitative results are shown in Figure []:

We implemented our model in PyTorch (Paszke et al., 2017) and trained using the Adam (Kingma and Ba, 2014) optimizer with a learning rate of 0*.*0001 and a batch size of 64. Due to time limit we only finished 1 epoch training both and it is enough for the purpose of making comparison between different word embeddings. Further full training will be done and different ways of combining distributional and definitional word vectors are to be implemented in future.

Quantitative results are presented in Table 3. We included three metrics we measured from the results of the NMT task in the purpose to make a comparison between our HybridVec and GloVe only word embeddings, They are train/evaluation accuracy, perplexity and BLEU.

# Discussion

5.1 Intrinsic Evaluation

5.2 Extrinsic Evaluations

# Conclusion

# References

1. 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-1)
2. 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-2)