# Best of both worlds “HybridVecs”: Hybrid distributional and Definitional word vectors

Haiyuan Mei
hmei0411@stanford.edu

Ranjani Iyer
iyerr@stanford.edu

## 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 attempt to show that a combination of distributional and definitional word vectors produced from an autoencoder provide an improvement for Neural Machine Translation. We found limited success in our autoencoder approach to capturing definitional contexts but that did not translate to a downstream extrinsic task.

## 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. These word vectors are unable to handle out-ofvocabulary (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).

## 2 Related Work

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.

## 3 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.

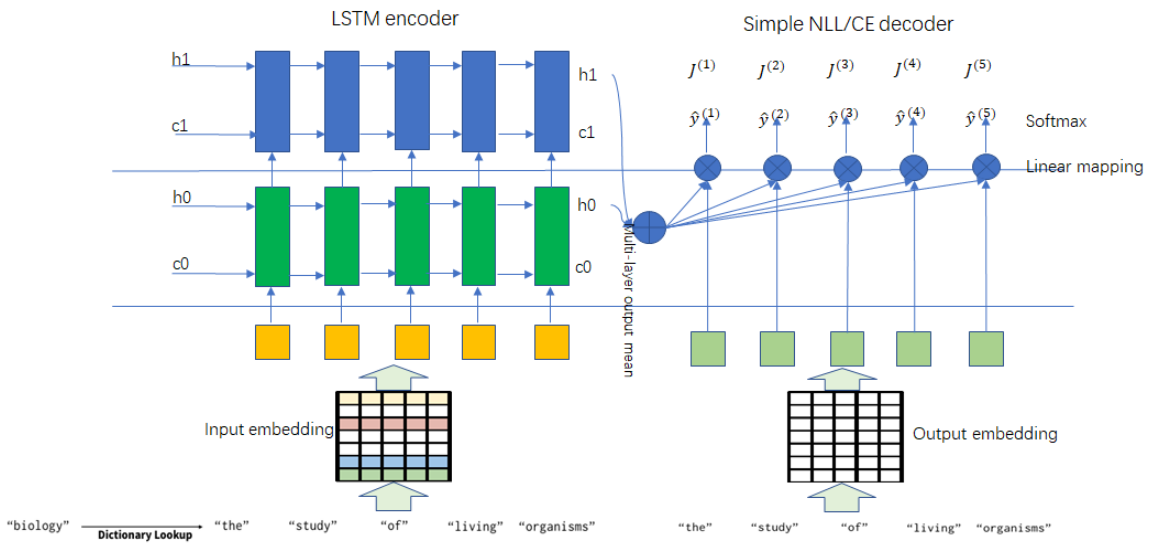


Figure 1: LSTM baseline model has an LSTM encoder and a simple conditional Language Model decoder learned in the way of normal classification method, which is softmax + Cross Entropy

## 4 Models

### 4.1 LSTM baseline model

This LSTM baseline model contains two word embeddings separately for encoder and decoder each. Let $V^D$ be the set of all words that are used in definitions, and $V^K$ be the set of all words that are to be defined. $V^D$ and $V^K$ are not necessarily the same but will be from the same vocabulary set. The definition of each word w from $V^K$ is a list of words from $V^D$ denoted as $d = (d_1, d_2, \ldots, d_T)$ where $d_T$ is the index of a word in vocabulary $V^D$. The definition for $d$ 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 $|V^D| \times$m matrix with the $i^{th}$ row an m-dimentional input embedding for the $i^{th}$ word in $V^D$. Depends how many layers we pass to the model, the last hidden state will be the same number of m-dimensional definition embeddings as the number of layers. The model is depicted in Figure 1, and the hidden layer can be described as the following equation, in which $E$ is our input embedding.

$$h = f_{E,\theta}(d) = LSTM_{E,\theta}(d)$$

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 learnt in the way of normal classification methods, using softmax, $|V^D|$ dimensional one-hot-vector and Cross-Entropy loss. For each definition of d in defs(w), the Cross Entropy is given by:

$$J(d) = \sum_t \log\left(softmax(\tilde{E}h + b)_{d_t}\right)$$

The total loss of all word definitions including multiple meanings of the same word is just the negative of sum over all sentences; it can also be interpreted Negative Log-Likelihood Loss NLL:

$$J_r(E, \theta, \tilde{E}) = -\sum_{w \in V^K} \sum_{d \in defs(w)} J(d)$$

We made the input and output embeddings different 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.

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 L2 norm between the predicted word embeddings and the learnt word embeddings, which gives the final loss function as:

$$J(E, \theta, \tilde{E}) = J_r(E, \theta, \tilde{E}) + \lambda \sum_{w,d} \left|\left|E_w - f_{E,\theta}(d)\right|\right|_2^2$$

$E_w$ denotes the input embedding associated with word w. If we make the penalty $\lambda$ a large number then after optimization we will end up having $E_w$ very close to $f_{E,\theta}(d)$ in Euclidean distance, which makes the definitional word vector hold very similar meaning to the defined word itself.

## 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 without attention to produce the dense representation h that represents the definitional embedding. In the decoder part another 2-layer LSTM LM is applied, and the training loss is to minimizes the negative log-likelihood between the predicted definitional word $\hat{d}$ 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 and relationships of the words in the definitions. We only evaluated this model intrinsically for its lack of evidence in representing the word meaning effectively.

## 4.3 Variational Autoencoder (VAE)

We also explored the fancier 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)

## 4.4 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 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.

## 5 Experiments

### 5.1 Data

For definitions, we follow the practice of previous work and employ data from the WordNet database (Miller, 1995). For the LSTM baseline model and Seq2seq model, we use the 400k vocabulary version of GloVe trained on Wikimedia 2014 and Gigaword 6 (Pennington et al., 2014) with 300 dimensional word vectors. These 400k words were used as input key for which we search in WordNet for definitions. Then the definitions were run through the two models where the hidden state between the encoder and the decoder was used to represent the input word definitional embedding. 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 default OpenNMT demo 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

Both the LSTM baseline and Seq2seq models are trained with the word vector dimension 300, and hidden layer dimension 150 such that they are comparable to GloVe 300d word vectors. 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.

### 5.3 Intrinsic evaluation

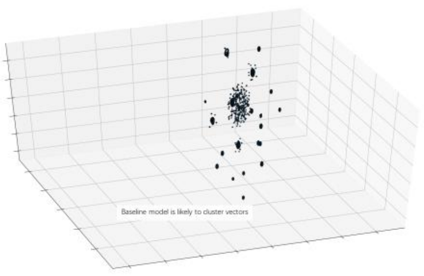
**Similarity[1] and Relatedness[2]**: We evaluate the quality of the embeddings produced from our autoencoder models by using a third-party word embedding benchmark test toolsets: Word Embedding Benchmark(WEB)[ ]. WEB is focused on evaluating and reporting results on common benchmarks (analogy, similarity 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

| | BLESS | ESSLI_1a | MEN | MTurk | RG65 | SL999 | WS353 |
|---|---|---|---|---|---|---|---|
| Glove | 0.82 | 0.75 | 0.737465 | 0.633182 | 0.769525 | 0.3705004 | 0.543326 |
| Baseline glove | 0.55 | 0.659091 | 0.51071 | 0.4226407 | 0.656402 | 0.3678366 | 0.449105 |
| Baseline rand | 0.52 | 0.613636 | 0.447908 | 0.3181051 | 0.6444908 | 0.3288122 | 0.35609 |
| S2S enc mean | 0.275 | 0.522727 | 0.106169 | 0.1370724 | 0.0890822 | -0.018433 | 0.051959 |

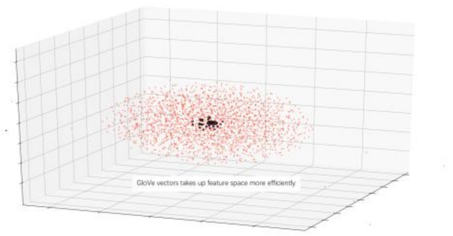
Table 1: Spearman's $\rho \times 100$ on various benchmarks. (GloVe: for GloVe vectors; Baseline glove: for LSTM baseline model initialized from GloVe; Baseline rand: for LSTM baseline model initialized randomly; s2s enc mean: for Seq2seq model with encoder output mean as the def vec.)

[1] SimLex999 [Hill et al., 2016] and SimLex333

[2] RG [Rubenstein and Goodenough, 1965], WS353 [Finkelstein et al., 2001], SCWS Huang et al. [2012] and MTurk Radinsky et al., [2011] Halawi et al. [2012].



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



tSNE for GloVe: GloVe makes use of feature more efficiently, grasp more sutle meaning of words

calculated in order to measure the ranks is Spearman's $\rho \times 100$.

Quantitatively Word Embeddings Benchmarks for GloVe, LSTM Baseline and Seq2seq model in table 1 reveals to us that our LSTM baseline model is roughly at the level of distributional method; more specifically the LSTM baseline trained by initializing from the GloVe vectors result in a better score compared to initializing randomly; but none could exceed the ground truth GloVe vectors. The more complex Seq2seq SAE model on the other hand the shows very limited evidence of such capability in matching GloVe distributed word representation.

Qualitatively we also explored through the 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. This is probably because word definitions, mainly from dictionaries, are more rigorously defined and are lack of the variation in the sense of expressing real world. One example is that definitional vectors are unable to capture different types of texts, unable to describe clichés and idioms. Future studies need to probably train them from a broader text source, especially different types of texts should be included. The Glove embeddings on the other hand makes use of feature space more efficiently, which means GloVe vectors can grasp more subtle meanings of words in real word.

## 5.4 Extrinsic Evaluation

The purpose of extrinsic evaluations in our work is to verify how useful definitional vectors are for downstream tasks. In order to compare different performance impacts of GloVe and HybridVec in downstream NMT tasks, we use the same GloVe 400K words as input vocabulary to generate embeddings for all the extrinsic evaluations. We train a translation model with OpenNMT-py on two different corpuses, first we apply the 10K default OpenNMT demo English-German corpus

| | No pretrained | Baseline | GloVe | HybridVec | description |
|---|---|---|---|---|---|
| **Train PPL** | 7.47 | 6.69 | 4.84 | 4.4 | 10k nmp demo sentence trained 1 epoch, with/o pretrained word vectors. Glove has most positive impact, LSTM baseline also exhibites positive impact |
| **Train ACC** | 56.29 | 58.4 | 64.32 | 66.1 | |
| **BLEU** | 0.93 | 1.37 | 1.99 | - | 10K nmt training demo 10 epochs, eval on 3k nmt val sentences,similar result as above perplexity and accuracy |

Table 2: Compare performance impacts on NMT task using LSTM baseline vector, GloVe and HybridVe. Note that '-' for HybridVec is done with Yandex 1M corpus and shown in table 3.

and 3k validation sentences to make a quick observation on different pre-trained word embeddings, the quantitative results is shown in [Table 2].

We finally come to the point of comparing NMT performance impacts between HybridVec and GloVe embeddings by applying the Yandex 1M English-Russian Corpus which has one million aligned English and Russian sentences (Yandex, 2018). The validation during training is done by

| | GloVe | HybridVec |
|---|---|---|
| **Train PPL** | 39.11 | 33.81 |
| **Train ACC** | 38.99 | 40.53 |
| **Val PPL** | 67.4823 | 67.188 |
| **Val ACC** | 35.9683 | 35.97 |
| **BLEU** | 5.01 | 4.69 |

Table 3: Compare performance improvements using GloVe and HybridVec (a combination of Glove and LSTM baseline vectors), trained with Yandex 1M corpus. Note that this is trained for only 185,000 steps since one of our deep learning platforms keep reporting segment faults memory problem (We make use of two deep learning environment, one on cloud and one locally, both with 8G GPU and over 26G RAM). The comparison is done both at step 185,000.

10% of the whole corpus, the final validation is done by 5000 sentences as suggested by OpenNMT system. Two pretrained embeddings are applied for the NMT task: (1) with GloVe vectors, (2) with HybridVec vectors combining both GloVe and LSTM baseline vectors, GloVe vectors come first. Quantitative results are shown in [Table 3].

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.

## 6 Conclusion

The concept of definitional embeddings is interesting because of the potential added intrinsic semantic meaning to the information captured by more traditional, distributional-based word embeddings. We aimed to explore whether encoding this different source of definitional information could add to the representational power of our current methods of embedding words.

Our experiments showed that our autoencoder approach is intrinsically successful at constructing word embeddings that have reasonable performance on similarity, relatedness, antonymy and compressing the definitions. In particular, examining recreated definitions showed that our learned compressed definitional embeddings were able to somewhat recreate the input definitions, and often captured the right vocabulary even if the words were out of order.

However, examining the t-SNE visualizations showed that the learned definitional embeddings for the SAE model do not evenly cover the vector space into which they're embedded in, suggesting that similarity learned between synonym sets isn't uniquely indicative. Further, average performance on our extrinsic evaluation neural machine translation task indicates that these definitional embeddings are not successful at capturing additional information. This suggests that the methods we employed for capturing definitional contexts do not capture the intended meaning that could provide additional value to distributional embeddings.

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 and then only used a vocabulary size of 50,000. 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. A future approach is to combine the definitional and distributional embeddings in a different manner. One such approach would be to generate the definitional embeddings with an auxiliary loss term that minimizes $L_2$ distance to the corresponding GloVe embedding for that word instead of doing a simple concatenation. We also want to explore getting definitions from multiple sources so that the models have more information that they can learn the definitional embedding space from. Lastly, our results indicate that our autoencoder approach is not able to compress interesting definitional contexts into embeddings and so we would like to explore other methods of doing so.

## 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.

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

Manaal Faruqui and Chris Dyer. 2015. Nondistributional 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.

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*.

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 Schutze. 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.