HybridVec: Hybrid Distributional and Definitional Word Vectors

Ranjani Iyer
iyerr@stanford.edu

Haiyuan Mei
hmei0411@stanford.edu

Abstract

Word vectors are typically computed by implementing distributional statistics, but these word vectors cannot represent unknown words. The ability to integrate word definitions with distributional statistics to create hybrid word vectors has the potential to improve performance on out-of-vocabulary tasks. A baseline bag-of-words and sequence-to-sequence auto-encoder were first iterated upon to obtain definitional word vectors that capture complementary information to distributional word vectors. Use of a sentence variational auto-encoder to compute word embeddings was also explored. Preliminary results suggest that a combination of distributional vectors (GloVE embeddings) and definitional word vectors produced from an autoencoder provide an improvement for Neural Machine Translation and warrants further testing.

1 Introduction

Word vectors are typically calculated as distributional statistics, (Mikolov et al., 2013) (Pennington et al., 2014) such as co-occurrence bag of word predictions, but the most logical source of words' meanings - dictionaries - are not leveraged in the process. We want to investigate the ability to use word definitions in the process of forming definitional word vectors that can work in parallel with existing distributional vectors to form a hybrid word vector.

We take word definitions as inputs into various autoencoder models (discussed in section 3) and use the hidden layer of the autoencoders as dense word embeddings of the definition. These embeddings are used to test against standard benchmarks to understand the information captured and can then be combined with existing distributional word vectors in varying degrees to improve the performance of downstream NLP tasks.

2 Related Work

There have been a number of prior attempts at deriving word vectors from dictionary definitions. Neural models using dictionary definitions and character-level morphology (Bahdanau et al. 2017), applying embedded definitions to a reverse lookup task (Hill et al. 2015), a skip-gram model (Tissier et al. 2017), and combining language modeling with derived definition embeddings (Noraset et al. 2016), are some examples of published work in the space.

This project is a continuation of previous work completed by Andrey Kurenlov, Tony Duan, Aneesh Pappu, and Rohun Saxena at Stanford university. Prior work began by building Def2Vec, a mapping of embedded word defintions into a semantically meaningful space, comparable to that of pretrained GloVe embeddings. The Def2Vec team demonstrated the utility of Def2Vec in improving the performance of a Neural Machine Translation model when the pre-trained vectors vocabulary is limited.

However, The Def2Vec team realized that definitional vectors alone are unable to perform as well as the highly robust distributional vectors in the NMT system; this motivated the introduction of combined distributional and definitional word vectors - Hybrid Distributional and Definitional Word Vectors. Including both types of representation can capture different aspects of a given word's meaning and the integrated performance may outperform either individual model.

Recently, Tom Bosc and Pascal Vincent published their successful results on training a Consistency Penalized Auto Encoder (CPAE) to capture semantic similarity better than distributional and current definitional word vectors. The model uses a LSTM to process the definition of a word and create a word embedding, which is then used by the decoder to reconstruct a bag-of-words representation of the definition.

3 Methods

As a continuation of previous work, we initially 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); and a normal Seq2seq Sentence Autoencoder model with both encoder and decoder as configurable recurrent neural network. After obtaining different word embeddings separately, the intrinsic evaluation is completed as a series of word embeddings benchmarks, comparing the LSTM baseline model, Seq2seq model, and existing GloVe word embeddings.

We finally applied our learnt word embeddings in combination with pretrained GloVe vectors to form our HybridVec embeddings and evaluated using OpenNMT as our extrinsic evaluation system.

We also explored the possibility of utilizing a more advanced Variational Autoencoder (VAE) model in creating definitional embeddings. The sentence VAE is based off Bowman et al. 2016 and is an RNN-based variational autoencoder generative model that incorporates distributed latent representations of entire sentences (definitions in our case). This factorization allows it to explicitly model holistic properties of sentences such as style, topic, and high-level syntactic features.

Models

3.1 LSTM baseline model

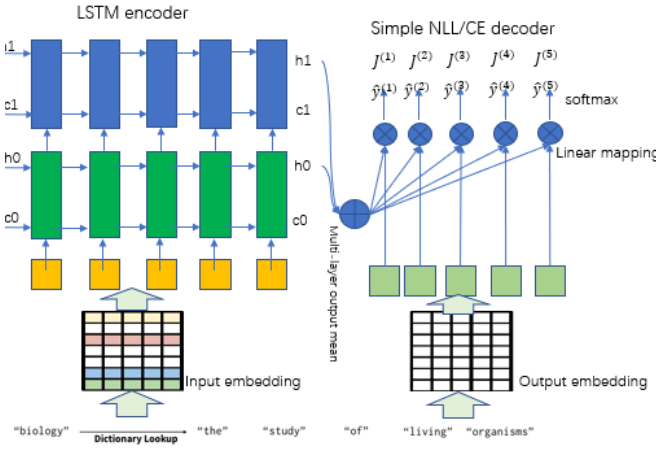

Figure 1. LSTM baseline model

This LSTM baseline model contains two separate word embeddings for both the encoder and the decoder. 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= $(d_1, d_2,..., d_T)$ where d_t is the index of a word in vocabulary V^D. The definition for d is 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 $|V^D|$×m matrix with the i^{th} row an m-dimentional input embedding for the i^{th} word in V^D. The last hidden state will be the same number of m-dimensional definition embeddings as the number of layers passed into the model. 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 from the encoder, and the sequence of output word embeddings corresponding to the word definitions. It is a simple conditional language model, where each predicted word is learnt through 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:

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

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

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

In our approach, we used different input and output embeddings, which could have caused the model to overfit. A unique word embedding matrix is an alternative way of implementation which can be explored in future experiments.

In order to minimize the distance between the definitional embeddings and the learnt word embeddings, a penalty weighted by λ is applied on the L2 norm between the predicted word embeddings and the learnt word embeddings, which gives the final loss function as:

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

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

3.2 Seq2seq Autoencoder

The second model we explored to create word embeddings takes the form of a Seq2seq autoencoder (SAE) 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 run through a 2-layer LSTM encoder without attention to produce the dense representation *h* that represents the definitional embedding. In the decoder, another 2-layer LSTM is applied. The training loss minimizes the negative log-likelihood between the predicted definitional

	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 ρ × 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 S2s model with encoder output mean as the def vec.)

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 because of its poor capability in representing the word meaning effectively.

3.3 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.

3.4 Variational Auto Encoder

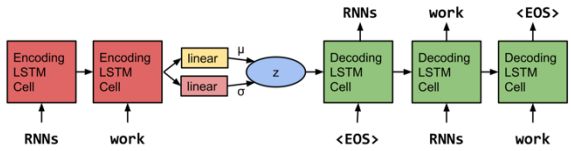

Figure 2. Variational Auto Encoder

We also tried implementing a variational auto encoder based on Bowman et al 2016, as shown in **Figure 2**, but were not able to complete testing and evaluation of the model. The model was meant to use single-layer LSTM RNNs in the encoder and the decoder, with the Gaussian prior acting as a regularizer on the hidden code. The poor results of the Seq2Seq model and the improved results of the CPAE suggest that more work needs to be done on refining the definitional word embeddings, and the generative VAE model with latent encoding of variables could provide improved results with continued testing and evaluation.

4 Experiments

4.1 Data

For definitions, we incorporated the datasets of previous work and employed data from the WordNet database (Miller, 1995). For the LSTM baseline model and Seq2seq model, we used 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 definitional words, for which we extracted definitions from WordNet. The definitions were then run through the baseline and seq2seq models, where the hidden state between the encoder and the decoder was used to represent the input word's definitional embedding.

Lastly, for the NMT task we used 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 functions well in providing faster comparisons between our HybridVec and GloVe embeddings.

4.2 Training

Both the LSTM baseline and Seq2seq models are trained with the word vector dimension of 300, and hidden layer dimension of 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 training was consistent with the prior work done in evaluating definitional and hybrid word vectors.

4.3 Intrinsic evaluation

Similarity and Relatedness: We evaluate the quality of the embeddings produced from our autoencoder models by using a third-party word embedding benchmark test toolset: Word Embedding Benchmark (WEB) [https://github.com/kudkudak/word-embeddings-

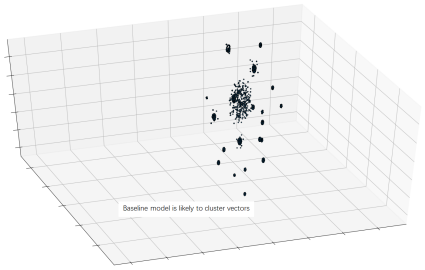

Figure 3. tSNE for LSTM baseline vectors shows that baseline vectors tend to cluster in feature space.

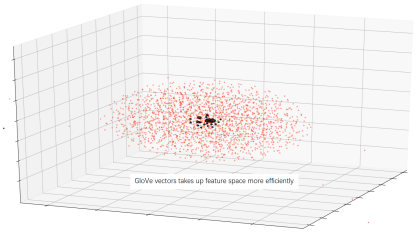

Figure 4. tSNE for GloVe: GloVe makes use of feature space more efficiently, grasping more subtle meanings of words

benchmarks]. WEB is focused on evaluating and reporting results on common benchmarks, such as 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 ranks are measured using the Spearman's $\rho \times 100$ metric.

Quantitatively, the Word Embeddings Benchmarks for GloVe, LSTM Baseline, and Seq2seq model in **Table 1** reveal that our LSTM baseline model is approaching similarity levels that are produced by distributional vectors. More specifically, the LSTM baseline trained by initializing from the GloVe vectors result in a better score than one that is initialized randomly. However, none of the tested models surpassed the ground truth GloVe vectors. The more complex SAE model, on the other hand, shows very limited evidence of capability in matching the GloVe distributed word representation.

Using t-SNE visualizations (van der Maaten and Hinton, 2008) of the test set embedding space allowed us to qualitatively explore the model performance. The predicted LSTM baseline definitional word embeddings tended to cluster in the feature space to a much smaller number of groups, when compared to vocabulary size. This is probably because word definitions, mainly from dictionaries, are more rigorously defined and lack the variation that exists in example corpora; definitional vectors are unable to capture different types of texts, describe clichés and idioms, or pick up on context clues. Future studies should train definitions from a broader text source, and try and incorporate different types of texts. The Glove embeddings, however, make use of feature space more efficiently, suggesting GloVe vectors can grasp more subtle meanings of words in the real word.

4.4 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, we used the same GloVe 400K words as input vocabulary to generate embeddings for all the extrinsic evaluations. We trained a translation model with OpenNMT-py on two different corpora: the 10K default OpenNMT demo English-German corpus, and the 3k validation sentences to make a quick observation on different pre-trained word embeddings. These quantitative results are shown in **Table 2**.

We compared NMT performance impacts between HybridVec and GloVe embeddings on the Yandex 1M English-Russian Corpus, which has one million aligned English and Russian sentences (Yandex, 2018). The validation was done on 10% of the whole corpus and it can be seen from **Table 3** that the validation perplexity during training is not ideal for such a large validation set. The final validation is done by 5000 sentences as suggested by OpenNMT system. Two pretrained embeddings are applied for the NMT task: one with GloVe vectors, and one with HybridVec vectors combining both GloVe and LSTM baseline vectors. 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

	No retrained	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: Comparing performance impacts on NMT task using LSTM baseline vector, GloVe and HybridVec. Note that BLEU for HybridVec is done with Yandex 1M corpus and shown in table 3.

	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: Comparing 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 kept reporting segmentation fault memory problems. We used two deep learning environment, one on cloud and one locally, both with 8G GPU and over 26G RAM. Both comparisons were at step 185,000.

HybridVec and GloVe only word embeddings: train/evaluation accuracy, perplexity and BLEU.

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 and lack of computation resources, we only managed to finish 185000 sentences in the training for both and it presents an initial comparison between different word embeddings. Further full training to convergence will be done and different ways of combining distributional and definitional word vectors are to be implemented in the future.

5 Discussion

Using different input and output embeddings is a possible cause of overfitting within our models; a unique word embedding matrix is an alternative to be explored in future experiments. More time would also allow for the ability to finish testing the VAE model. Word definitions, mainly generated from dictionaries, are more rigorously defined and lack the variation required to capture real word semantic information. Future studies are needed to use a broader text source in the training. Due to lack of computation resources, the extrinsic training has not fully converged, so the full exploration of HybridVec impact on downstream tasks is not complete.

6 Conclusion

Definitional embeddings are fascinating, although rarely studied by researchers, for their potential in capturing complementary semantic information when combined with traditional distributional word embeddings. We aimed to explore whether encoding this different source of information could add to the representational power of current methods in embedding words.

Our experiments showed that an LSTM baseline autoencoder approach is intrinsically successful at constructing word embeddings at a level roughly similar to distributional embeddings. However, examining the t-SNE visualizations showed that the learned definitional embeddings for the LSTM baseline model tend to cluster in a small number of groups, suggesting insufficiency in representing more subtle word meanings in natural language. Furthermore, by observing validation datasets in intrinsic evaluation, it is apparent that the overall feature space of definitional word vectors is much smaller than GloVe. Further study on these clusters is to be explored in future work.

Extrinsic evaluation on both the OpenNMT demo corpus and Yandex 1m corpus suggests that during training, the perplexity and accuracy are both better than those of GloVe. However, the final evaluation on 5000 validation sentences diverges. The BLEU metric of our HybridVec vector is of a lower value for LSTM baseline vectors. Given both the positive and negative impact on perplexity and BLEU, and the fact that none of the extrinsic experiments are trained to convergence, the full utility of HybridVec remains to be seen.

7 Acknowledgements

Special thanks to Andrey Kurenkov for general mentorship and guidance throughout this project, as well as helping review our code and documents, and thanks to the Stanford CS229 teaching staff.

8 Contributions

As a group working on this collaborated project, we contributed equally overall. Haiyuan Mei is responsible for improving the LSTM baseline model and Seq2seq model by contributing to the existing code base, and getting the test results regarding the two models. Ranjani Iyer is responsible for the initial model understanding and working on implementation of the VAE model.

Code base: https://github.com/andreykurenkov/HybridVec

9 References

1. Kurenkov A., Duan T. (2018). Def2Vec: Learning Word Vectors from Definitions. Stanford Class Project

2. Pappu, A., Saxena, R. (2018). Best of both worlds ”HybridVecs”: Combining distributional and definitional word vectors. Stanford Class Project

3. Tissier, J., Gravier, C., & Habrard, A. (2017). Dict2vec : Learning Word Embeddings using Lexical Dictionaries. *EMNLP*.

4. Bosc, T.and Vincent, P. (2018). Research.fb.com. Available at: https://research.fb.com/wp-content/uploads/2018/10/Auto-Encoding-Dictionary-Definitions-into-Consistent-Word-Embeddings.pdf? [Accessed 14 Dec. 2018].

5. Bowman, S., Vilnis, L., Vinyals, O., Dai, A. M., Jozefowicz, R., & Bengio, S. (2016). Generating Sentences from a Continuous Space. In *Proceedings of the Twentieth Conference on Computational Natural Language Learning (CoNLL).*

6. 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.

7. Manaal Faruqui and Chris Dyer. 2015. Nondistributional Word Vector Representations. *arXiv:1506.05230 [cs]*. ArXiv: 1506.05230.

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

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

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

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

12. 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.

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

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

15. 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.

16. 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.

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

18. 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.

19. 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.

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

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

22. 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].

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