Best of both worlds ”HybridVecs”: Combining distributional and definitional word vectors

Aneesh Pappu
apappu@stanford.edu

Rohun Saxena
rohuns@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-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).

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 will try to 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 definitional 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. We support this hypothesis with intrinsic and extrinsic evaluation.

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. 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) .

The main paper we focus on that looks at how definitional contexts can be used in improving word embeddings is in Bosc and Vincent. The paper looks at a simple autoencoder model to create embeddings, where the encoder tries to compress a defintion with a standard LSTM and the decoder is a fully connected layer over the entire vocabulary plus a softmax layer trying to decode the definition from the compressed representation. The loss function is the negative log probability summed over each word in the definition according to their predicted unigram probability from the softmax layer, which inherently models each definition as a bag of words model. The paper found that this approach alone created embeddings with some meaning based on intrinsic similarity evaluations, but these were only useful in low-data settings because they didnt perform as well on evaluation metrics as GloVe did.

3 Methods

3.1 Autoencoders

In order to create definitional embeddings, we use the general autoencoding framework to learn definitional embeddings that represent the definitions of the associated word. The general premise of an autoencoder is to use a deep architecture to learn two mappings – one, a forward mapping of the input x to some $f(x) = h$, and the second, an inverse mapping of h back to the original$x$, as visualized in Figure 2. The output h is a dense representation that, if the model is trained well, theoreticlaly is a good representation of the input data. We apply this autoencoding framework and use recurrent neural architectures to create dense representations of input definitions, which we then use as our newly created definitional embeddings.

3.2 Bag-of-Words autoencoder

For our initial definition encoding baseline mode, we trained a recurrent LSTM model to encode our definitions such that we maximize the negative log-likelihood for the words in the definition upon passing the encoded hidden state through a fully connected softmax layer across the entire vocabulary (based off of Bosc and Vincent) as depicted in Figure 3. In detail, for each word w in the training corpus, we lookup the definition $d(w)$. For each definitional word d in said definition, we pass the embedding for d (input embeddings trained from scratch) through the LSTM to create a dense representation h that represents our definitional embedding.

$$\boldsymbol{h = f_{E,\theta}(d) = \mathrm{LSTM}_{E,\theta}(d)}$$

Figure 1: The encoder consists of an LSTM to produce the output definition embedding h

The decoder network is a fully connected layer across the entire vocabulary, with the objective function being to minimize the negative log-likelihood probability for every word in the definition. This objective essentially takes the form of a multi-class, multi-label classification problem: where the inner softmax computes the probability for a single word across the entire vocabulary, h is our definitional embedding, and E is the output

$$\sum_t \log p_{\bar{E},b}(d_t|h) = \sum_t \log\left(\text{softmax}(\bar{E}h + b)_{d_t}\right)$$

fully connected layer decoder.

When summed over the entire corpus of words and their definitions, results in an overall objective function of

$$J_r(E,\theta,\bar{E}) = -\sum_{w\in V^K}\sum_{d\in\text{defs}(w)} \log p_{\bar{E},b}(d|h = f_{E,\theta}(d))$$

And additionally, we add a form of regularization to bring the encoded representation w closer to its input embedding through using an L2 loss term, resulting in an overall cost function of

$$J(E,\theta,\bar{E}) = J_r(E,\theta,\bar{E}) + \lambda \sum_{w\in V^D\cap V^K}\sum_{d\in\text{defs}(w)} ||E_w - f_{E,\theta}(d)||^2$$

Importantly, this model assumes that the definitions are modeled as a bag-of-words with the goal of maximizing the unigram probabilities for each word in the definition (because of the singular softmax across the entire vocabulary of the decoder).

3.3 Sentence Autoencoder

Our second, more complex autoencoder takes the form of a sentence autoencoder (SAE) model that respects the initial syntactic structure of the sentence as opposed optimizing for a bag-of-words prediction as the previous baseline model. Given an input word *w*, we look up its definition $d(w)$. Similar to our baseline, 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. As opposed to the previous bag-of-words assumptions, the sentence autoencoder model 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/relationships of the words in the definition as shown in Figure 4. It's important to also note that this model is *not* trained with attention as a classic sequence to sequence model might be trained – knowledge of the previous hidden states used in encoding, as required by attention, would allow the Seq2Seq model to cheat and copy over the hidden state from the encoding, thereby not actually learning a dense definitional representation *h*.

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.

4 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 Bag-of-Words and Sentence Autoencoder model employs a hidden state of size 100 and 200 respectively. 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.

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

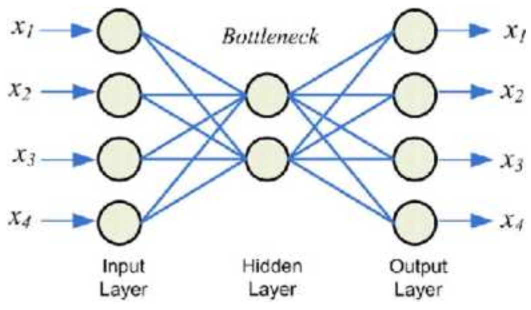

Figure 2: Template layout of an autoencoder, where the central bottleneck hidden layer is the dense, learned representation of the input

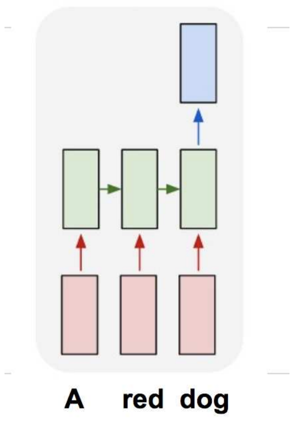

Figure 3: Diagram of bag-of-words autoencoding model

4.3 Intrinsic evaluation

Similarity/Relatedness: 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, r_s, which ranges between -1 and $+1$ (Table 1). An r_s of $+1$ indicates high correlation and means that the embeddings are similar and related, and a r_s of -1 implies poor performance on similarity and relatedness tasks.

Antonymy: We also assess the quality of the embeddings produced from our autoencoder models using an antonymy metric to measure how far antonyms are in the embedding space, a task distributional embeddings struggle on. We created the set of antonyms by going through the synsets of the words in our vocabulary and finding their antonyms using WordNet. We used cosine similarity to measure how similar any two embeddings are in space. A cosine similarity of $+1$ indicates that the embeddings are identical and a similarity of 0 indicates the vectors are far apart in space. Results for the antonym evaluations are in Table 2

Qualitative: The first qualitative evaluation is looking at the pairs of input and output definitions to make sure that the encoder models are accurately able to compress and then expand back out to the input definition. Below is a random selection of words with their input defintion and the predicted definition from the SAE model.

```
associations
input: a formal organization of people
or groups of people
predicted: <SOS> organization point
of people or groups of <EOS>

acceptance
input: the state of being acceptable
and accepted
predicted: <SOS> of of being acceptable
accepted accepted <EOS>

lengthy
input: relatively long in <UNK> <UNK>
protracted
```

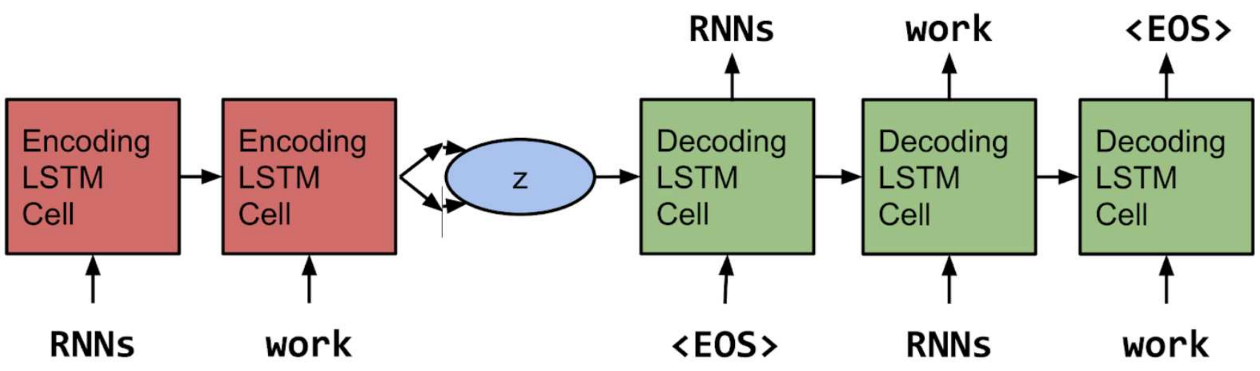

Figure 4: Diagram of sentence autoencoding model

Model	SimLex999	RW	WS353	WS353S	WS353R	MEN	MTurk	RG65
GloVe	0.2975	0.3142	0.477	0.6035	0.415	0.6809	0.6193	0.6762
BOW	0.054	0.0378	0.0418	0.0825	-0.0443	-0.0083	0.0762	0.0689
SAE	0.0854	0.102	0.125	0.1724	0.0728	0.0971	0.1987	0.3041

Table 1: Spearman Correlations for GloVe, Bag-of-Words, and SAE embeddings on similarity and relatedness datasets.

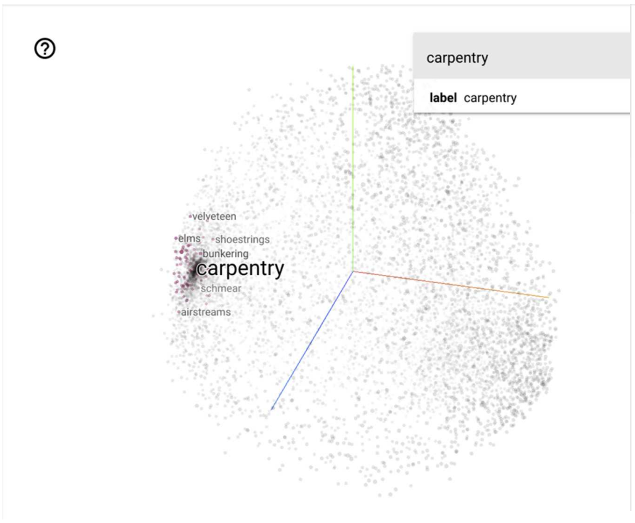

Figure 5: BOW embeddings t-SNE visualization

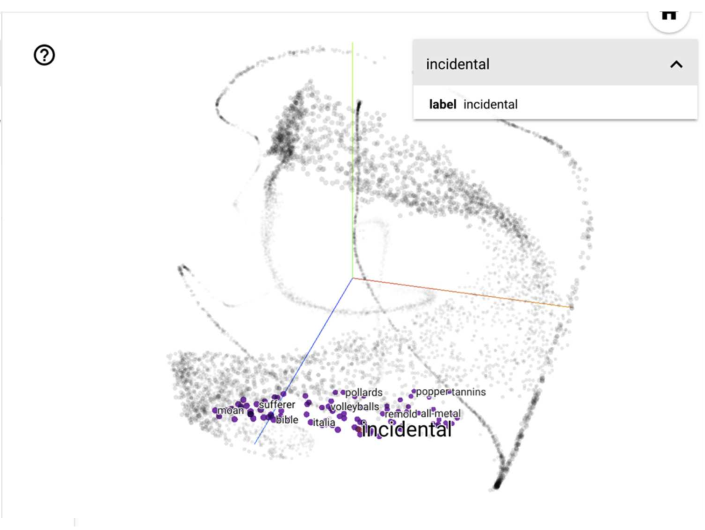

Figure 6: SAE embeddings t-SNE visualization

```
predicted: <SOS> in in <UNK> <UNK>
exercising <EOS>
```

We also qualitatively verify through t-SNE visualizations (van der Maaten and Hinton, 2008) the embedding space that our predicted Bag-of-words and SAE models is one that is reasonable for word embeddings (Figure 5 and 6). These visualizations represent where embeddings for our full vocabulary lie in space to make sure that our embedding space is smooth and spread out as would be expected for word embeddings.

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

Model	Cosine Similarity
GloVe	0.64
BOW	0.58
SAE	0.65

Table 2: Average cosine similarity for GloVe, Bag-of-Words, and SAE embeddings over all antonym pairs.

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.

5 Discussion

5.1 Intrinsic Evaluation

Similarity/Relatedness: Our initial intrinsic evaluations on similarity and relatedness show that the definitional embeddings generated by the two autoencoders are able to model some level of semantic similarity between words that are similar. For example on the RG65 test we get a nontrivial r_s correlation of 0.3 compared to GloVe at 0.67 on that dataset. While definitional embeddings do not capture as much similarity information as GloVe embeddings, the definitional embeddings do model some similarity and there may be added value when combined with GloVe.

Antonymy: The results from our experiments on average cosine similarity show that the embeddings produced by our Bag-Of-Words autoencoder had a 10% lower average cosine similarity than standard GloVe Embeddings, which is positive in that it shows these new embeddings better separate words that have opposite meanings. Antonymy was one of the shortcomings of distributional word vectors so it is a positive sign that on the Bag-of-Words model words that are antonyms have embeddings that are dissimilar. However, the trend of better antonymy evaluations did not translate to the Sentence Autoencoder model, as it performed just about the same as GloVe embeddings do on the task. This may be related to the clustered t-SNE visualizations, covered in more detail later. However, the fact that antonyms are as dissimilar as GloVe given how clustered all the embeddings are might suggest that the distributional vectors are still separating the antonyms well but since everything is clustered near each other the cosine similarities are close to each other.

Qualitative: Examining the autoencoding ability of our models; the sample word definitions and their recreations in Figure 4.3 were produced by our SAE model. We can see that the reconstructed definition for associations has some of the right words but theyre out of order. This trend is true for the majority of the examples and so our hidden representations are retaining some of the definitional meaning encoded which is a good sign that the models are creating embeddings that are representing definitional meanings.

Looking at the t-SNE visualizations (Figure 5 and 6) for the Bag-of-Words model the distribution of embeddings is fairly even in the space which is a good sign since we expect 400,000 words with a variety of meanings to occupy most of the embeddings space. However, the embeddings generated from the SAE model appear to be more clustered and less evenly distributed. These results are a little concerning because we expect word embeddings to be distributed evenly but that is not the case with the embeddings generated from the SAE model. This suggests any synonymy or antonymy relations learned aren't necessarily differentially unique / could be 'false positives' in a way, since the emebeddings aren't evenly spread out in the vector space.

5.2 Extrinsic Evaluations

Finally, looking at the performance of the new embeddings created by the BOW model and the SAE model concatenated with GloVe, we see in Figure 3 that the performance in accuracy and per-

Embeddings	Perplexities	Accuracy
GloVe	35.60	39.55
GloVe + BOW	35.55	39.62
GloVe + SAE	36.30	39.41

Table 3: Quantitative evaluation of embeddings on NMT task.

plexities was not affected by the new embeddings. Both GloVe + BOW and GloVe + SAE performed just as well as GloVe alone, showing that the new definitional embeddings generated from concatenating the BOW and SAE representations with the original GloVe vectors did not add any additional value over GloVe embeddings.

6 Conclusion

The concept of definitional embeddings are 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. 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.

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 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.