Towards Universal Sentence Embeddings

Sruthi (15190)
gorantlas@iisc.ac.in

Nilesh (15007)
anilesh@iisc.ac.in

Lokesh (14355)
lokeshn@iisc.ac.in

Abstract

The tremendous success of word embeddings raises the obvious question if similar methods could be used to derive improved embeddings for word sequences like sentences, paragraphs and documents as well. Sentence embeddings, specifically, can also be treated as building blocks for paragraph and document embeddings as they capture the semantics of each sentence. In this project, we aim at finding generalized sentence embeddings and use these representations on various downstream tasks. We then try to compare our results with the existing state-of-the-art results on these tasks. Further, we plan to provide a detailed analysis about the models which capture the semantic and syntactic information of the sentences.

1 Introduction

1.1 Motivation

In the field of Natural Language Processing, representation of the input data plays a key role. Natural language is a sequence of words which do not have a pre-determined set of features. These sequence of words are not immediately understandable by the traditional machine learning models. To solve this problem, various supervised, unsupervised and semi-supervised techniques have been proposed to generate sentence representations (i.e., numerical low dimensional representation of the sentence) that can be fed as input to the machine learning models.

With the invent of word embeddings, the world moved on to generate sentence embeddings using bag-of-words representation in the beginning. Later, techniques like concatenation, summation, dot product of word embeddings also have shown good performances. However, neural network based models have outperformed these methods as they have much better representation power. Some of the recent works towards this area are, (Kim, 2014) which used CNN on top of word embeddings to generate sentence embeddings, Skip-thought vectors (Kiros et al., 2015) which has described an approach for unsupervised learning of a generic, distributed sentence encoder etc.

One of the very recent works by (Arora et al., 2017) has described an unsupervised non neural network technique that outperformed the sophisticated models like RNN's and LSTM's In this method, they use word embeddings computed using one of the popular methods on unlabeled corpus like Wikipedia, represent the sentence by a weighted average of the word vectors, and then modify them a bit using PCA/SVD. This weighting is shown to improve the performance by 10-30% in textual similarity tasks. As can be clearly seen, this method is very simple to implement while giving good performance. Some experiments performed in this paper show that, this method which ignores word order must be much better at exploiting the semantics than RNNs and LSTMs. This forms the motivation for our project. We aim at exploiting the advantages of this model and LSTMs in a hope to generate better sentence embeddings.

1.2 Problem Statement

Our work is mainly focused on generating generalized sentence embeddings. To this end, we first compare the standard models with the method proposed by (Arora et al., 2017) to establish the conclusions that have been made. We then propose a new architecture that exploits the advantages of the approach in this paper and also the LSTM and RNN based models. We use these sentence embeddings to perform SNLI textual entailment task and compare results with the existing state-of-the-art. In order to study the reasons for the better performance of the ensemble model, we perform detailed analysis on the architectures in various settings.

In the remainder of the report we discuss about our strategy and experimentation in detail.

2 Related Work

We discuss existing models which have been proposed to construct sentence embeddings. While there is a large body of works in this direction several among these using labelled dataset to obtain sentence embeddings in a supervised manner (Wieting et al., 2015) (Conneau et al., 2017) while other focus on unsupervised, task-independent models (Kiros et al., 2015) (Arora et al., 2017) .

Skip-thought (Kiros et al., 2015) tries to reconstruct the surrounding sentences from surrounded one and treats the hidden parameters as their vector representations while **InferSent** (Conneau et al., 2017) uses BiLSTM sentence encoder model using supervised labelled dataset - SNLI to generate universal sentence embedding. **Paragram-Phrase** (Wieting et al., 2015) uses Paraphrase database to generate sentence embedding in a supervised manner using simple averaging of word vectors. While (Wieting et al., 2015) implemented 6 more neural network architecture in paper but Paragram-Phrase model beats all the NN model such as LSTM.

The directed inspiration for our work is (Arora et al., 2017) which learned universal sentence embeddings by using simple weighted word averaging of the word vectors in the sentence and then remove the projections of the average vectors on their first principal component (common component removal). We aim at incorporating word order by using composition of weighted averaging approach with LSTMs in a hope to see a performance improvement.

3 Method

3.1 SIF

(Arora et al., 2017) represent the sentence as a weighted average of the words in the sentence. The weights are defined by a smooth inverse frequency term (SIF). SIF is $\frac{a}{a+p(w)}$ where a is a user defined smoothness hyperparameter and $p(w)$ is the probability of occurrence of the word giving less importance to the more frequent words. Apart from the word averaging, we learn the principal component of the corpus which represents the vector direction of the most frequent words. We subtract the principal component from the weighted average so that the resultant vector has a high potential to act as a **discourse vector** for the sentence. This discourse vector gives the information about what is being talked about in the sentence. The base paper provides a mathematical justification regarding the correctness of SIF.They have also done an extensive analysis of performance on various datasets against sophisticated models and found that this simple strategy delivers comparable performance. Having fetched the discourse vectors for each sentence, we can use them for a variety of front end tasks.

3.2 BOW vs SIF

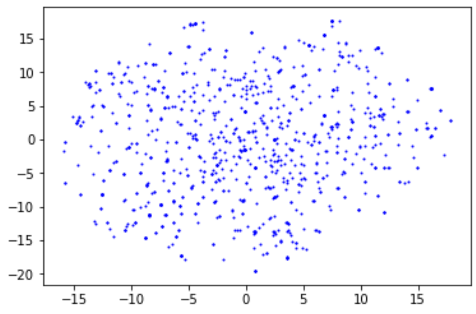
BOW refers to bag of words model. Here the representation of the sentence is treated as a simple word averaging. Hence the final embeddings of the sentence is **order invariant**. This kind of representation fails to capture the negation of a sentence etc. The SIF model, though it is also order invariant, is superior to the BOW because it tries to extract what is called a **Discourse vector,** C_s from the sentence. By suppressing the syntactic words like the, is are etc. and thereby projecting just the important semantic words, this representation performs way better.We have results to elucidate this claim and is presented in the later sections of the paper.

Figure 1 showcases the sentence vectors learnt through SIF model on a t-sne reduced dimensional space. The first figure shows the overall distribution of the sentences in the SNLI corpus. The second figure elucidates how the similar, contradicting, and neutral sentence co occur in the representations. One interesting thing to observe is that the representations of the contradicting sentences are close. Especially the contradicting sentences 14a, 14b are very close because they differ only by the word ”not”.

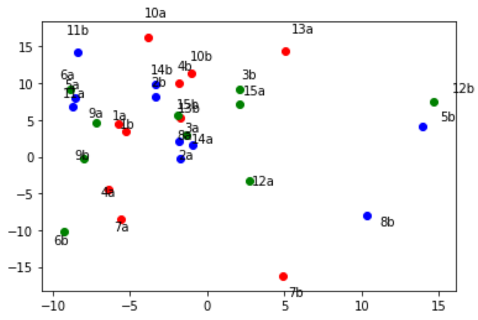
We did analysis on performance of the SIF vectors and BOW vectors as is, and found many interesting things. We took the SNLI dataset which is a three class classification problem with the classes being entailment, contradiction and neutral and observed the following.

Feature	BOW	SIF
Similarity measure is Cosine similarity	33.3%	42.3%
Similarity measure is dot Product	33.3%	35.02%
Number of sentence pairs with cosine similarity = 1	209	0

Table 1: Analysis of BOW and SIF embeddings



(a) Distribution of the sentences in the corpus



(b) Distribution of similar, contradicting and neutral sentence pairs

Figure 1: Visualization of projected SIF vectors on t-sne reduced dimensions

First and second rows of Table 1 shows the accuracy of BOW and SIF embeddings on classification. For classification, we took the similarity measure as cosine similarity and dot product similarity. In each of the schemes, we regarded the highest valued pair as entailment, second highest as neutral and the least as contradiction. What is interesting is in the third row where BOW vectors give cosine similarity as 1 for most of the pairs and this observation can be treated as a empirical proof for the fact that principal component subtraction results in discourse vector extraction.

3.3 Ensemble Approach

We propose three ensemble approaches, LSTM_SIF, LSTM_AVG_SIF and BiLSTM_MAX_SIF.

3.3.1 LSTM_SIF

As shown in Figure 2, our ensemble model is based on concatenation of the LSTM output with the SIF embedding for the given sentence. We propose a supervised learning approach to learn the sentence embeddings. Consider the the SNLI task which is technically a three class classification task. Given a *premise* S_1 and a *hypothesis* S_2, the model has to predict if the hypothesis is an entailment or contradiction or neither to the premise, in which case, its labelled as neutral. Let $S_i = [w_{i,1}, w_{i,2}, \dots, w_{i,N}]$, $i \in \{1, 2\}$ (All the sentences are padded to a length N). We use the glove vectors (Pennington et al., 2014) of 300 dimensions as word embeddings to all the words which is fed as input to the LSTM at every time step $t = \{1, 2, \dots, N\}$ is given by:

$$x_{i,t} = e_{i,t} = glove(w_{i,t})$$

The final hidden vector representation is calculated as follows:

$$\begin{aligned}
f_t &= \sigma(W_f x_t + U_f h_{t-1} + b_f) \\
i_t &= \sigma(W_i x_t + U_i h_{t-1} + b_i) \\
o_t &= \sigma(W_o x_t + U_o h_{t-1} + b_o) \\
c_t &= f_t \circ c_{t-1} + i_t \circ \sigma(W_c x_t + U_c h_{t-1} + b_c) \\
h_t &= o_t \circ \sigma(c_t)
\end{aligned}$$

$h_{1,N}$ and $h_{2,N} \in \mathbb{R}^d$ are the final hidden vector representation of the sentences. This is taken as embedding to the sentence after passing the words through the LSTM because it will have the information about all the words in the sentence in the sequential order. We call them v_{β_1} and v_{β_2} respectively.

v_{α_1} and $v_{\alpha_2} \in \mathbb{R}^{300}$ are the corresponding representations of the sentences S_1 and S_2 obtained from SIF as below:

$$v_{\alpha_i} = \sum_{t=1}^{N} \frac{a}{a + p(w_{i,t})} e_{i,t}$$

where $e_{i,t} \in \mathbb{R}^{300}$ is obtained from the glove embeddings as above. We concatenate the vector representations obtained from both the models and give it as input to a fully connected layer with non-linear activation to reduce the dimensions and call them v_1 and v_2 respectively.

$$v_i = RELU(W_{dim}(v_{\alpha_i} \oplus v_{\beta_i}) + b_{dim})$$

Parameters
$W_f, U_f, W_o, U_o, W_c, U_c, W_i, U_i \in \mathbb{R}^{d' \times d}$ $b_f, b_o, b_c, b_i \in \mathbb{R}^{d'}$
$W_{dim} \in \mathbb{R}^{300 \times d}$ $b_{dim} \in \mathbb{R}^{300}$
$W_v \in \mathbb{R}^{3 \times 300}$ $b_v \in \mathbb{R}^3$

Table 2: List of parameters

where $\oplus$ represents the concatenation symbol.

We then concatenate v_1 and v_2 and give it as input to a three class classifier:

$$\begin{aligned} v &= v_1 \oplus v_2 \\ v' &= RELU(W_v(v) + b_v) \\ o &= softmax(v') \\ y &= argmax(o) \end{aligned}$$

Where y represents the predicted output class.

Let t_j represented the target output class of the jth sentence pair out of the total n input sentence pairs and y_j represents its predicted output class. Then we calculate the cross-entropy loss as follows:

$$L(\theta) = -\frac{1}{n} \sum_{j=1}^{n} t_j \log y_j$$

We train this classifier using Adam optimizer to minimize the loss mentioned above.

3.3.2 LSTM_AVG_SIF

In this model, instead of considering the last hidden vector representation $h_{i,N}$, we take the average of all the hidden vector representations at each time step. Then the vector v_{β_i} becomes:

$$v_{\beta_i} = \frac{1}{N} \sum_{t=1}^{N} h_{i,t}$$

The results in table 4 show that LSTM_AVG_SIF than LSTM_SIF.

Automated Metric Entailment task is a three class classification task. Hence we use accuracy measure to compare the performance of various models.

$$accuracy = \frac{tp + tn}{tp + tn + fp + fn} \quad (1)$$

Hyperparameters	
a	10^{-3}
d	300
d'	300
batch size	128
optimizer	Adam
learning rate	10^{-4}
epochs	15-20

Table 3: List of hyperparameters

Results		
	Validation	Test
LSTM	0.38	0.385
LSTM_SIF	0.51	0.52
LSTM_AVG	0.54	0.53
LSTM_AVG_SIF	0.58	0.58
BiLSTM_MAX_SIF	0.86	0.85

Table 4: The table shows the accuracy rate of the SNLI classification task using different models.

3.3.3 BiLSTM_MAX_SIF

As shown in Figure 2, our ensemble model is based on concatenation of the LSTM output with the SIF embedding for the given sentence. In this model, we concatenate output of forward LSTM $h_{i_f,t}$ and backward LSTM $h_{i_b,t}$ at each timestep $t = \{1, 2, \ldots, N\}$ to get $h_{i,t}$. We will do max pooling over $h_{i,t}$ to get v_{β_i}. Then the vector v_{β_i} becomes:

$$\begin{aligned} h_{i,t} &= h_{i_f,t} \circ h_{i_b,t} \quad \forall t = \{1 \ldots, N\} \\ v_{\beta_i} &= \max_{1 \le j \le n = 300} h_{i,t_j} \quad \forall t = \{1, \ldots, N\} \end{aligned}$$

We concatenate the vector representations obtained from SIF v_{α_i} with v_{β_i} and give it as input to a fully connected layer with non-linear activation to reduce the dimensions and call them v_1 and v_2 respectively.

$$v_i = RELU(W_{dim}(v_{\alpha_i} \oplus v_{\beta_i}) + b_{dim})$$

4 Results

In table 4 we show the accuracy results obtained using different models:

From the results it can be clearly seen that LSTM_SIF model is showing 13-14 points improvement in the accuracy. Hence we conclude that, as claimed in (Arora et al., 2017), the SIF

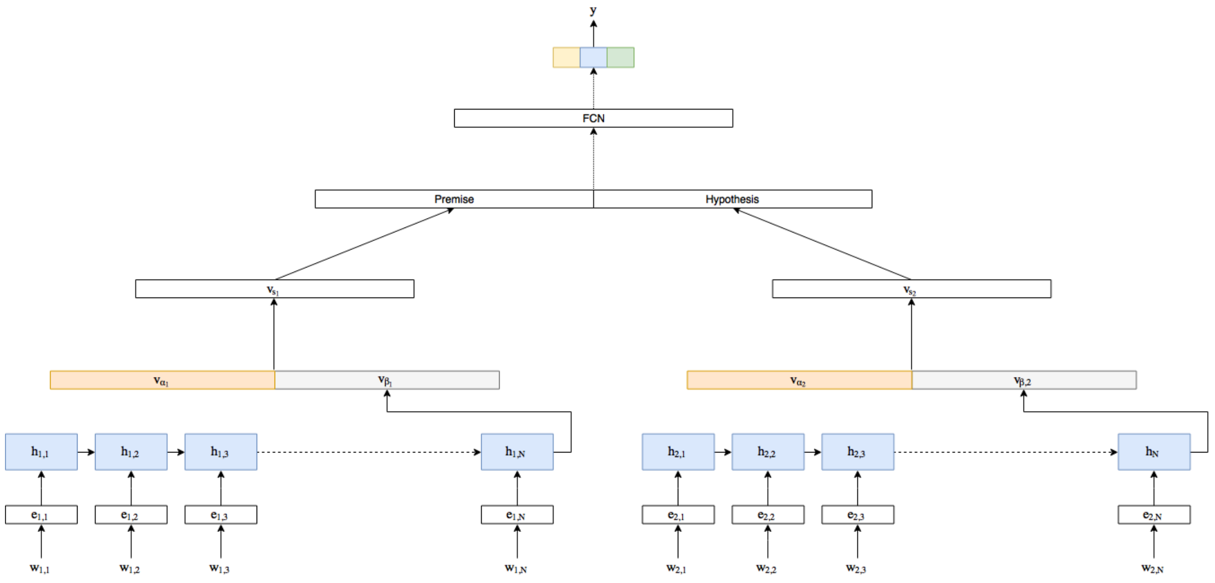


Figure 2: Proposed model on SNLI entailment task. The premise S_1 is given as input to the first LSTM and the hypothesis S_2 is given as input to the second LSTM. Each sentence $S_i, i \in 1, 2$ is tokenized into words $\{w_{i,1}, w_{i,2}, ..., w_{i,N}\}$ and the corresponding word embeddings are extracted, $\{e_{i,1}, e_{i,2}, ..., e_{i,N}\}$. These embeddings are fed as input to the LSTM. The last hidden state from the LSTM $h_{i,N}$ forms the vector v_{β_i} which is concatenated with the vector obtained from the weighted average method v_{α_i}. These concatenated representations of both the sentences are then concatenated and given as input to a fully connected layer and then to a 3-way softmax

embeddings are actually capturing the semantic information of the sentences rather than the simple LSTM. In order to incorporate this information in the basic LSTM, we modelled the LSTM_AVG in expectation that it will capture both semantic and sequential information. As can be seen, this model tends to reach the expectations by again improving the accuracy by 16 points. One reason LSTM_AVG is performing better than LSTM_SIF is that it's not just a weighted average of the independent words but the vectors that we are performing average on also contain sequential information from the beginning of the sentence till that word. However, LSTM_AVG_SIF shows nearly 20 points improvement in the accuracy standing out as the best model.

We perform an analysis on the LSTM model by shuffling the input words in the given sentence randomly and giving as input to the LSTM_AVG model and LSTM_AVG_SIF model in order to check if the sequential information is being captured by the LSTM. The results are shown in table 5

The results clearly show that the scrambled input to both the LSTM models degrades the performance. Hence we can conclude that LSTM is capturing the sequential information.

Results on scrambled input		
	Validation	Test
LSTM_AVG	0.50	0.49
LSTM_AVG_SIF	0.54	0.53

Table 5: Shuffled words as input to LSTM models

5 Transfer Task

We learn the SIF embeddings from SNLI corpus and use them on many other datasets. We do this to check how well the embeddings generalize across different datasets. We do this with the belief that the direction of the principal components on any datasets tend to be the same. Eg. in SNLI corpus the direction of the first principal component was towards the words like *just, when, even, one, up, little, way, there, while, but etc*. It is easy to see that these words don't add much information to the discourse vector not only in the SNLI corpus but in any corpus in general. We ensemble the SIF embeddings with many other sophisticated models to get generalized sentence embeddings. To check if this ensemble really helps, we tried with the same configuration of the model with and without SIF embeddings and were able to observe a consistent improvement in performance with the ensemble models. The results are presented in the Results section 6.

For the transfer tasks, we fetch the sentence embedding from ensemble model and used it to get the sentence embedding for a transfer task. In the transfer task models we didn't update the sentence embedding which we have got from ensemble model i.e. we didn't perform end-to-end learning as it gives as better insight how the sentence embedding performs when used on different datasets if we would have updated the sentence embedding vector while training for transfer task using backpropogation it would have given us some increase in accuracy on transfer task model. We performed transfer task to evaluate our sentence embedding on 14 different datasets. Transfer tasks model are simple model such as Binary Classifier (MR,CR, SUBJ, MPQA), N Class Classifier (SST2, SST5, SICK-E, SNLI) and Relatednes Score Classifier (SICK-R, STS12, STS13, STS14, STS15, STS16, STS-Benchmark)

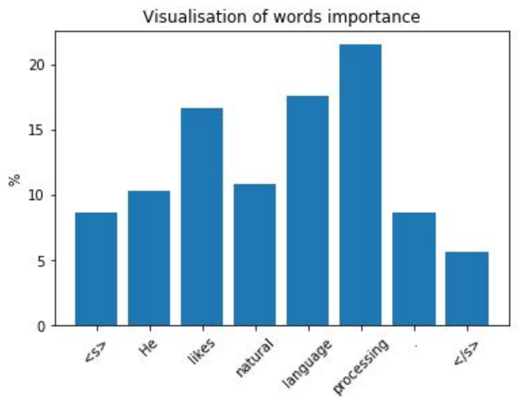
6 Datasets and Analysis

6.1 Dataset

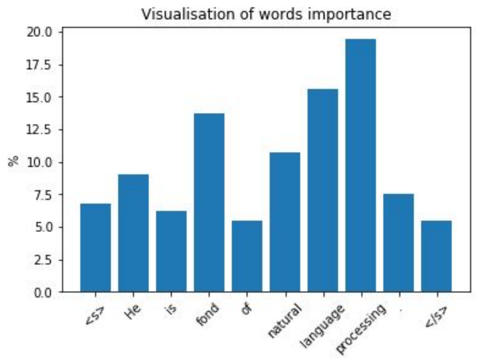
To train different models for generating sentence embeddings - SIF, BiLSTM+Max, BiLSTM+Max+SIF we have used SNLI dataset to generate it. And then used the generated sentence embedding from these model to get the sentence embedding for 14 different datasets which we have used as part of our evaluation of sentence embedding on transfer tasks. The 14 different datasets we used are MR (Movie Review), CR (Customer Review), MPQA, SUBJ, SST2, SST5, SICK-E, SICK-R, STS-12, STS-13, STS-14, STS-15, STS-16, STS-Benchmark.

6.2 Sentence Visualization

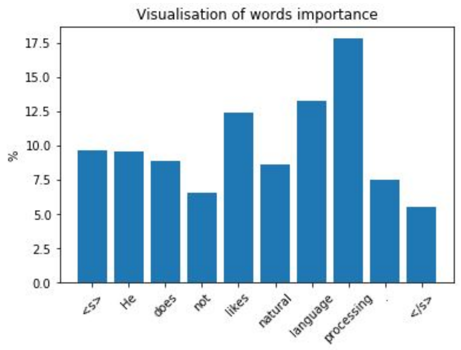
Figure 3 represents the visualization of three sentence embeddings obtained from BiLstm+SIF ensemble modelviz.
1. He likes Natural Language Processing
2. He is fond of Natural Language Processing
3. He does not like Natural Language Processing
The bar chart plot represents the weightage given to each word in the sentence. We can see from the charts that with addition of SIF vectors the context words are given more importance and other syntactic words are subdued.



(a) Distribution of the sentences in the corpus



(b) Distribution of similar, contradicting and neutral sentence pairs



(c) Distribution of similar, contradicting and neutral sentence pairs

Figure 3: Visualization of projected SIF vectors on t-sne reduced dimensions

DataSetName	BOW	SIF	BiLSTM+Max	BiLSTM+Max+SIF
Movie Review(MR)	77.23	78.45	80.05	79.83
Customer Review(CR)	78.22	78.0	84.61	86.19
MPQA	87.87	88.23	90.18	90.22
SUBJ	91.18	90.87	92.19	94.68
SST2	80.29	82.23	81.77	83.4
SST5	44.66	41.32	42.44	43.5
SICK-E	78.55	84.32	86.3	85.8
SICK-R	79.92	79.84	88.55	91.22
STS12	53.2	54.57	59.36	62.23
STS13	50.75	49.96	58.44	57.86
STS14	55.64	58.9	67.95	71.39
STS15	59.25	62.23	70.25	68.6
STS16	57.91	61.82	70.18	70.34
STS Benchmark	64.74	68.45	75.76	77.16
SNLI	65.98	72.0	84.5	85.27

Table 6: Transfer Task Results

7 Evaluation

In table 6 we show the accuracy results obtained on different transfer tasks (datasets) using BOW, SIF , BiLSTM_MAX and BiLSTM_MAX_SIF models. It is clearly evident from the table SIF beats BOW model as expected and our model BiLSTM_MAX_SIF also beats BiLSTM_MAX (Arora et al., 2017). Hence we can conclude empirically that the SIF embeddings, by suppressing the lesser significant details and capturing only what is requires help the front end models perform better.

8 Conclusion

In this project we conclude that the claim made by (Arora et al., 2017) that a simple model of averaging word vectors of a sentence using smoothed inverse frequency is itself enough to capture the semantic information of the entire sentence. However, our results also show that LSTMs are good at capturing the sequential information of the sentence. Hence our effort to combine both the models in order to get advantage of sequential as well as semantic information is successful and is evident in the accuracy results obtained on the SNLI entailment task.

The transfer tasks also show that these embeddings serve as good generalization of the representations of the sentences.

References

Sanjeev Arora, Yingyu Liang, and Tengyu Ma. 2017. A simple but tough-to-beat baseline for sentence embeddings.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. *CoRR*, abs/1705.02364.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1746–1751.

Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S. Zemel, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. 2015. Skip-thought vectors. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'15, pages 3294–3302, Cambridge, MA, USA. MIT Press.

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.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2015. Towards universal paraphrastic sentence embeddings. *CoRR*, abs/1511.08198.