Extending the Scope of Co-occurrence Embedding

Jinhong Mi, Yuetong Wang, Jiren Zhu
Stanford University
{jmi1997, yuetong, jirenz}@stanford.edu

Abstract

Motivated by the success of co-occurrence based embedding methods, we intended to extend the scope of embedding beyond word level. We used GloVe to obtain co-occurrence statistic based embedding for phrases and words jointly and evaluate them on word analogy, word similarity and sentiment analysis. Results showed that co-occurrence based methods generalizes well to phrases, even improving word-level structures on word similarity tasks as phrases are added to vocabulary. These phrase vectors allowed us to study word-level compositional structures directly. They were integrated into Deep Averaging Network and were shown to improve performance on the sentiment analysis task like word vectors do. Exploration of further extending the scope of embedding methods to synsets and POS-tags is also discussed in appendix.

1 Introduction

1.1 Overview

In natural language processing tasks, word embedding techniques have shown to capture semantic and syntactic information of natural language and improve performance of various down stream tasks, including sentiment analysis [6], parsing [15] and named entity recognition [13].

While these embeddings greatly improve the performance of downstream models, they still suffer from some intrinsic disadvantages. On the side of properly understanding language, words are not necessarily atomic units of language. In natural languages, multi-word phrases do not always represent the summation of their component words. For example, "take care of" and "look after" should have very similar representations, whereas it is not clear if word embeddings of "take" or "look" capture this information. It is also unlikely that downstream tasks will be able to infer this information from word level representations in its usually much smaller dataset.

We propose a model that generalizes the scope of co-occurrence based embedding techniques. We treat a natural language sentence as a continuous flow of possibly overlapping signals. The co-occurrence statistics of these signals encode the information about the language. We count the co-occurrence of these signals and factor the co-occurrence matrix into vectors using the currently state-of-the-art GloVe method [13].

More specifically, we tried two approaches. We extended embedding to n-grams and trained "gram vectors" on the latest Wikipedia dump corpus. These vectors have representation for words as well as for frequent n-grams. We also extended embedding to synsets and POS-tags and trained "combined-feature vectors" on a smaller dataset. We will be discussing the results of gram vectors in the main sections of this paper. Our results of combined-feature vectors are more exploratory and they will be presented in Appendix A.

1.2 Main results

Our experiments showed that learning embedding for word and n-grams together preserves and even improves the structures of word vectors, see Section 3.1, 3.2. These learned embeddings allow us to directly observe compositional relations between the vector of a phrase and vectors of component words, see Section 3.3. These trained gram vectors are shown to integrate well into the sentiment analysis task, see Section 3.4.

1.3 Organization of this Paper

In the rest of Sections 1 we will continue to discuss background, related work and why we choose co-occurrence factorization method, GloVe, to obtain embeddings. Sections 2 and 3 will discuss the work we did on using co-occurrence statistics to obtain embeddings for words and phrases. Section 2 focuses on how we obtained the gram vectors and Section 3 details all the evaluations we performed on these gram vectors. The beginning of Section 3 summarizes these experiments and lists information about relevant figures and tables. For the work and results on synset vectors, see Appendix A.

1.4 Background and Related Work

1.4.1 Word embedding

Word embedding techniques, gaining popularity with the work of Bengio et al. [1], assigns a vector representation for each word, the vector is then fed as input to downstream models. These vectors show interesting clustering structures. Recent advances have also shown that word vectors admit rich linear structures. The Skip-Gram and CBOW models by Mikolov et al. [10, 11, 12] showed that the learned word vectors have nice properties, such as “king” - “man” + “woman” = “queen”. They also constructed the word analogy dataset as a main metric to evaluate the structure of word vectors. Global Vector model, GloVe, by Pennington et al. [13] used global co-occurrence count to train vectors. They achieved even better results on word analogy and several other tasks. Using word representation has been shown to improve the performance of various down-stream tasks,including sentiment analysis [6], parsing [15] and named entity recognition [13].

1.5 Co-occurrence Methods, GloVe and Why We Chose Them

Co-occurrence methods assign embeddings to words, in our case also n-grams or synsets, such that the co-occurrence count of two word vectors over a global corpus is approximated by the dot product of their vectors. These methods leverage global statistical information of the corpus and enables unsupervised training on very large unlabeled datasets.

Early co-occurrence based models use singular value decomposition (SVD) to factorize the co-occurrence matrix. The GloVe model [13] improves upon previous co-occurrence models by factorizing the smoothed log co-occurrence count matrix with down sampling for frequent words. The logarithm allow linear operations on vectors to reflect ratios of co-occurrence count, thus GloVe vectors have very rich linear structures.

We did a preliminary experiment with raw co-occurrence count and found that co-occurrence encode very valuable information of the language (detailed can be found in Appendix B). This experiment led us to believe that successfully encoding co-occurrence of language features other than words would also show interesting results. It is than natural to use state-of-the-art method GloVe to factorize the co-occurrence matrix and obtain vectors.

1.6 Model Objective

We define a “signal” to be the appearance of a word, a phrase, a synset or a POS-tag. To each type of signal i we assign a vector representation v_i. The set of all signals has size $|V|$. For any two signals i and j, denote their co-occurrence count as X_{ij}. We want to encode the information of the huge matrix X with size $|V|^2$ with vectors v_i and v_j. Using the GloVe method [13], we want $v_i \cdot v_j$ to approximate $\log(X_{ij} + 1)$. Adding bias w_i for each signal, smoothing and sub-sampling and l^2

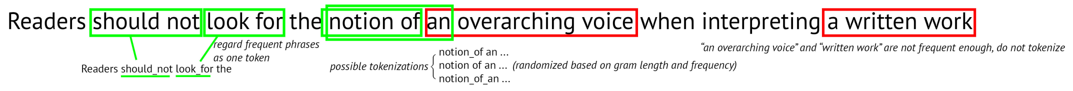
Readers should not look for the notion of an overarching voice when interpreting a written work
regard frequent phrases as one token
Readers should_not look_for the
possible tokenizations
notion_of an ...
notion of an ...
notion_of_an ...
(randomized based on gram length and frequency)
"an overarching voice" and "written work" are not frequent enough, do not tokenize

Figure 1: Example of Tokenization

loss, the final objective function to be minimized is

$$\mathcal{L} = \sum_{ij} f(X_{ij})(v_i \cdot v_j + w_i + w_j - \log(X_{ij} + 1))^2$$

We used the tools provided at https://nlp.stanford.edu/projects/glove/ to compute the vectors.

1.7 Related Work

Mikolov et al. [11] trained phrase vectors along with word vectors using the Skip-Gram model and showed that phrases like "Canada Air" can have its own embedding. Le and Mikolov [7] also obtained paragraph vectors to encode variable length text into fixed length vectors.

There has also been quite a few studies on encoding the information of a knowledge base. Socher et al. [3] used Neural Tensorial Networks to predict knowledge base relations by inner products of vectors. Rothe et. al [14] uses existing word embeddings and impose more restriction on the vectors based on the relations.

Our model regard co-occurrence count as the valuable information to encode in the vectors and directly count the co-occurrence of phrase and synsets with context words, (or context phrases and synsets).

2 Extending distributed representation to n-grams

In this section, we explore the possibility to extend the distributed representation to n-grams. We first extract the n-grams that are considered interesting from the raw text, based on purely statistical method. Then, with all n-grams and corresponding gram-counts (single words are considered as 1-gram, if necessary), we can build our tokenizer, which is a fundamental tool in our model and will be discussed in Section 2.3. This tokenizer takes a sentence as input, and output a series of tokens, each of which is a n-gram in our dictionary. In other words, if we define V to be the vocabulary including all n-grams that are considered interesting, this tokenizer translates each sentence in English into a sentence in our own vocabulary V. For example, consider the following sentence:

"According to the president of the united states"

Ideally, it may be tokenized as:

"[According to] [the president of] [the united states]"

However, other results are also possible given that the resulting tokens all belong to our vocabulary:

"[According to the] [president] [of the] [united states]"

Another example of this probabilistic tokenization method is shown in Figure 1.

2.1 Dataset

We used Feb. 2017 Wikipedai dump with 1.9B tokens as our training corpus. We chunked the datafile of size 11.5GB into 76 chunks of size 150M. The xml corpus is first sanitized by WikiExtractor at https://github.com/attardi/wikiextractor. The text is then transformed into lower case and non-alphabetical characters are removed.

2.2 Grams Count

We scan through the dataset chunk by chunk, and maintain the dictionary of all n-grams ($n \leq 3$) that appeared before, and their occurrence counts. For each gram s, we define the importance of s, $I(s)$ as:

$$I(s) = \frac{\#\text{occurrence of } s \text{ in the text}}{\min_{v, v \text{is a single word appears in s}} \#\text{occurrence of } v \text{ in the text}}$$

$I(s)$ normalizes the occurrence of a gram by its least frequent component. So although two frequent words may appear a lot and thus appear many times as a gram candidate, their I score is lower than that of a pair of words that always appears together.

After we process each chunk, we only keep the V grams with highest importance where V is the vocabulary size we care. In this specific experiment we used $V = 400{,}000$.

2.3 Tokenization

The Tokenizer, which partitions each sentences into a series of non-overlapping n-grams, uses a randomized heuristics that described as following:

(i) Maintain an iterator which points to the next unprocessed word in the sentence. At the beginning, this iterator points to the first word in the sentence.

(ii) Each time, we generate the length of the next token randomly. Let S_i denote the number of occurrence of the i-gram starting from the current iterator. For example, consider the sentence 'I hope that my petition could be approved' where the iterator points at the word 'petition', then S_3 denote the number of occurrence of the gram 'petition could be', in our training dataset. Then, we pick gram-length k proportional to some heuristics function $F(k, S_k)$. This function should encourage both longer grams and more frequent grams. In our implementation, we pick $F(k, S_k) = S_k \times w[k]$ where $w[1] = 1, w[2] = 10$ and $w[i] = w[i-1] + 3$ for all $i \geq 3$.

(iii) Continue this process until we tokenize the whole sentence. Obviously, we may get different results in different runs.

After tokenizing the Wikipedia dataset, we find that about 30% of the text is tokenized into phrases. The resulting text contains 1.4B tokens. Of the whole vocabulary of size 400,000, around 55% are words, 40% are bigrams and 5% are 3-grams.

2.4 Algorithm

We use GloVe [13] to factorize the co-occurrence statistics. After we tokenize the text into n-grams and phrases, we use the source code provided at https://nlp.stanford.edu/projects/glove/ to count co-occurrence and factor the co-occurrence matrix. We use the optimized hyper-parameters mentioned in the paper (window size 10, $x_{max} = 100$) since our dataset is very similar to the earlier Wikidump that the original paper used as corpus.

3 Evaluation of N-gram Vectors

We performed word analogy 3.1, word similarity 3.2, sentiment analysis 3.4 and visualized some phrase vectors using PCA 3.3. With the experiments of word analogy and word similarity, we intend to show that gram vectors nicely preserved the linear and cluster structures between words and even improved such structure, as reflected by the word similarity task. Then we will show that gram vectors allow us to study the compositional structure of words and grams through an example of PCA visualization. Finally, we will show that gram vectors can be easily utilized into downstream tasks like word vectors does, and get reliable performance with an example in sentiment analysis.

We list all figures involved in this section for easy reference. Word analogy results are shown in Table 1. Word similarity results are shown in Table 2. PCA visualization of vectors is shown in Figure 2. Sentiment analysis results are shown in Table 3.

Corpus	Method	Dim	vocab	coverage	sem 1	sem 10	syn 1	syn 10	tot 1	tot 10
1.4B	**Gram**	100	400,000	99.37	70.78	85.59	56.41	78.91	62.93	81.94
1.4B	**Gram**	300	400,000	99.37	78.98	**90.32**	58.60	82.41	67.85	86.00
1.6B	GloVe†	300	400,000	-	**80.8**	-	61.5	-	70.3	-
6B	Skip-Gram†	300	-	-	73.0	-	66.0	-	69.1	-
6B	CBOW†	300	-	-	63.6	-	**67.4**	-	65.7	-
6B	GloVe*	100	400,000	100.00	65.34	80.91	61.26	82.93	63.11	82.01
6B	GloVe*	300	400,000	100.00	77.4	89.92	67.00	**87.71**	**71.74**	**88.71**

Table 1: Word Analogy Results

Models in bold face are trained by our approach. **Gram** assigns vector representation to words and n-grams. GloVe†, Skip-Gram†, CBOW† are results reported in [13]. GloVe* are publicly available GloVe vectors, we did evaluation on them with our own test setup. Best performance in each category is in bold face. For details text Section 3.1

3.1 Word analogy

The word analogy dataset is constructed by Mikolov et al. [10] to study the linear structures of word representations. After tokenizing the text differently and adding a lot of n-grams into the vocabulary, one might worry that the original linear structures of the word vectors are no longer preserved. We employ the word analogy dataset to show that the structure of gram representations co-exist nicely with the structure of word representations.

3.1.1 Experiment Setup

We modified the script in the GloVe package to evaluate word analogy. Semantic analogies include capital of countries, currency, city in state and family members. Syntactic analogies include adjective to adverb, opposite, comparative, superlative, present-participle, nationality adjective, past tense, plural and plural verbs. The top answers are ordered by cosine similarity. We added top-10 accuracy to account for the newly added gram-vectors and do not perform any filtering on the vocabulary. The experiments were performed on 100 dimensional and 300 dimensional gram vectors. We compare to the results of GloVe, Skip-Gram and CBOW on the same tasks reported by [13] since we are using the same test setup as theirs. We also perform independent tests on publicly available GloVe.6B vectors to obtain both top 1 and top 10 similarity scores.

3.1.2 Results

The results are shown in Table 1 on page 5. We compare the performance of gram embeddings to GloVe, CBOW and Skip-Gram in the bottom section of the table. These embeddings are trained on very similar corpus, and the results are thus reliable. Embeddings with corpus size $1.xB$ are trained on Wikipedia data, embeddings with corpus size $6B$ are trained on Wikipedia + GigaWords data [13]. It can be seen that Gram vectors has performance comparable to GloVe vectors on word analogy tasks, especially performing well on semantic tasks. So we are convinced that introducing short phrases into the vocabulary of GloVe vectors is not disrupting the original word embedding structure.

It is also very straightforward to see that increasing dimensions for gram vectors significantly improves performance, as the model is allowed more flexibility to arrange the vectors.

Lastly, we observed that phrases do share linear structures with words, confirming the results mentioned in word2vec phrase embedding [11]. When given a country-capital pair, for example “Athens-Greece”, our model outputs “United Kingdom” to be the top match for “London”, (correct: “England”). So treating phrases as an atomic token allows the model to learn these composed names.

Model/ρ	GloVe*.100	**Gram**.100	GloVe*.300	**Gram**.300
MC-30	0.6271	0.6436	0.7026	**0.7061**
MEN-TR-3k	0.6809	0.7016	0.7375	**0.7468**
MTurk-287	0.6194	0.6673	0.6332	**0.6745**
MTurk-771	0.5805	0.6316	0.6501	**0.6766**
RG-65	0.6907	0.6993	**0.7662**	0.7649
RW-STANFORD	0.3664	0.3649	0.4118	**0.4407**
SIMLEX-999	0.2976	0.2922	**0.3705**	0.3614
VERB-143	0.3023	0.3993	0.3051	**0.4237**
WS-353-ALL	0.5290	0.6052	0.6054	**0.6691**
WS-353-REL	0.4955	0.5658	0.5726	**0.6413**
WS-353-SIM	0.6037	0.6910	0.6638	**0.7284**
YP-130	0.4543	0.4647	**0.5613**	0.5241

Table 2: Word Similarity Results

Best performance on each task is in bold face. **Gram** is trained by our approach. GloVe* are publicly available GloVe vectors trained on 6B corpus [13]. We used the evaluation scripts and datasets provided by [4]. See Section 3.2 for details.

3.2 Word similarity

Word similarity is another popular metric for evaluating the structure of word embeddings. We will show that gram vectors improves upon word vectors on this task.

3.2.1 Experiment setup

We used the tools provided by [4] to evaluate the Pearson correlation between the cosine similarity of gram embeddings and word similarity ranked by human. We compare our results trained on the 1.4B wiki-corpus with publicly available GloVe vectors trained on the 6B Wikipedia plus GigaWord corpus [13]. All tests has vocabulary covered near or at 100%, with the exception of RW-STANFORD dataset where GloVe vector vocabulary missed $12\%(252/2034)$ of the test vocabulary and gram vectors missed $31\%(628/2034)$. We use cosine-similarity between word vectors to rank word similarity. See Table 2 on page 6.

3.2.2 Results

We see that although trained on one fourth of GloVe's corpus size, gram vectors perform consistently better that GloVe vectors on multiple word similarity tasks. One of the most significant increase is on VERB-143 dataset ($\rho : 0.31 \to 0.42$) where word pairs are verbs like "calls-help". A possible reason is that the same verb in different phrases may have drastically different meanings depending on its neighboring prepositions. Allowing different representation for verb phrases allows the model to encode different meanings in different verb phrases. This, in turn, locates the verb vector itself in a more optimal location in the vector space. We shall show in the next section that we can inspect these possibilities by directly looking at word and phrase vectors at the same time.

3.3 Phrase Clustering

One direct advantage of training gram vectors is to obtain representation of phrases and their components at the same time. This allows us to investigate the compositional structures of word representations directly. We observed that a lot of frequent phrases are in form verb + preposition. One verb combined with different prepositions tend to have very versatile meanings. We chose a set of words and performed PCA on their vector representation to embed them into two dimensions. The vectors are 300 dimensional, trained on 1.4B Wikipedia corpus.

The center phrase is chosen to be "give up", labeled red in the figure. We selected a set of words and phrases that are synonyms of the center phrase, labeled blue. For all phrases involved we also add their word components and the sum and average of their components. See Figure 2 on page 7

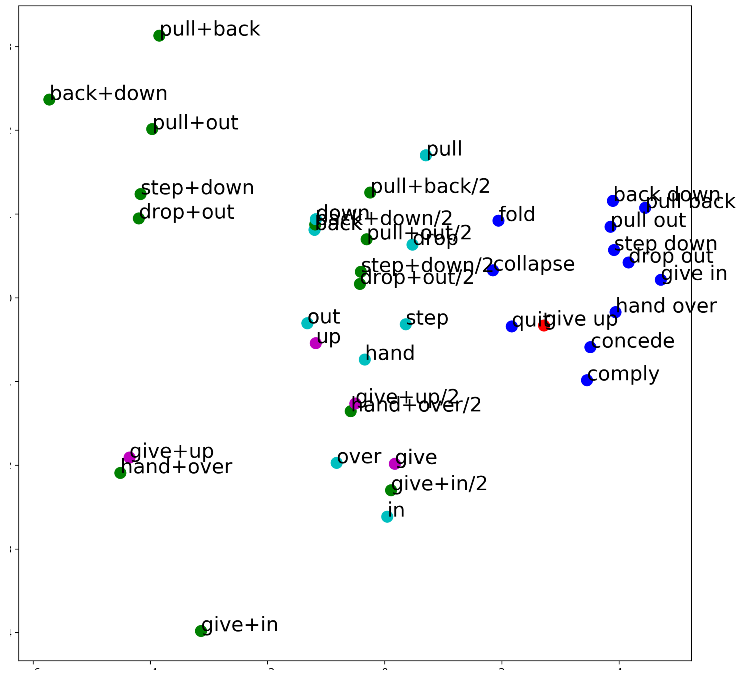

Figure 2: PCA Visualization: Synonyms of "give up"

The center phrase, "give up", is labeled red. Its synonyms are labeled dark blue. Embeddings for the components of the center phrase and their linear combination are labeled pink. Embeddings for components of synonym phrases are labeled light blue. Their linear combinations are labeled green. For details, see Section 3.3.

It is easy to see that the learned gram vectors of "give up" is much closer to the meaning "quit" than the component words "give" and "up", or their addition and average. This pattern repeats itself on other phrases. We can also see that phrases of similar meaning: "back down", "step down", "pull out" and "drop out" are clustered together like similar words do when trained on co-occurrence statistics. Training word vectors allows us to observe linear and clustering patterns of natural language. It is very exciting to see that training gram vectors further allows us to directly observe compositional structures of language, which is the core of a lot of natural language problems.

3.4 Sentiment analysis

In this section, we shall show that similar to word vectors, gram vectors can be integrated into downstream models very readily and offer performance gain just as word vectors do.

3.4.1 Experiment setup

We use Deep Averaging Network developed by Iyyer et al. [6] and run it on the Stanford Sentiment Tree Bank dataset [16] to evaluate to performance of our embedding technique.

Compared to using word vectors, we have one additional step of tokenizing sentences into phrases. For the training, development, and testing set from Stanford Sentiment Tree Bank, we apply our tokenizer to each sentence, to get the corresponding training, development, and testing set for our n-gram set-up where each sentence is a string of words or grams. When we use word tokenization, each word contributes its vector representation. When we use gram tokenization, each gram contributes its vector representation. These vectors are fed into a DAN and obtain the 5-class accuracy result.

We compare our models to randomly initialized vectors on both word and gram tokenization. We also compare our models to GloVe vectors [13] since DAN obtained its best result on sentiment analysis using GloVe vectors as well. Results are demonstrated in Table 3.

Dataset	Model	Tokenization	Dim	5-class Acc. (%)
-	random	word	100	41.8
-	random	gram	100	40.2
1.4B	**Gram**	gram	100	**42.2**
6B	GloVe	word	100	41.8
-	random	word	300	42.6
-	random	gram	300	40.0
1.4B	**Gram**	gram	300	43.8
6B	GloVe	word	300	44.8
640B	GloVe	word	300	**47.7**

Table 3: Using Gram Vectors on Sentiment Analysis

Bold face models are trained by our methods. The best result in each category is in bold face.
Word tokenization means that each word in the sentence contributes its corresponding embedding. Gram tokenization means that after being tokenized by our tokenizer, each phrase in the sentence contributes its embedding. For details see Section 3.4.

3.4.2 Results

We see from Table 3 that both 100 and 300 dimensional gram vectors imporves upon randomly initialized vectors on this task. Showing that we successfully encoded semantic and syntactic information into the n-grams.

We also see that 100 dimensional gram vectors achieves better sentiment scores than word level GloVe vectors trained on comparably sized corpus. Suggesting that gram vectors have competitive performance. In the case of 300-dimension vectors, the performance of gram vectors is half-way between the random embedding and the GloVe model on 6B dataset. This possibly comes from lack of training data and domain difference.

When vectors are 100 dimensional, limited amount of information can be encoded, so our model performs equally well compared to GloVe. As the dimension increases, more and more information are stored in the embeddings, and the lack of training data (1.4B vs 6B vs 640B) will hurt the performance of our model more.

Another important point to note is that, the dataset our model is trained on is relatively ‘biased’ compared to the sentiment analysis task itself. Material from Wikipedia is expected to have an objective stance, and that’s exactly the opposite of sentiment analysis. Therefore, it is highly likely that our model ignores the n-grams with strong emotion, simply because they rarely occur in the training data. We expect a boost in the classification accuracy if we could train our model on a more comprehensive dataset, say, common crawl. Comparatively, only one fourth of the corpus GloVe 6B vectors are trained on is Wikipedia, so it should suffer less from domain difference.

Overall this experiment is suggestive that these vectors can be well incorporated into existing models and improve their performance.

4 Conclusion

We extended existing global co-occurrence statistics methods to obtain embeddings for not only words but also phrases. We trained the phrase vectors on a billion token level corpus. Our embedding rivals existing word-level embeddings on word analogy tasks, showing good structure among embedding of words. It improves notably on word similarity tasks compared to purely word based embedding techniques, suggesting that adding phrases to the vocabulary may allow co-occurrence models to better obtain embeddings for words. We exhibited another advantage of phrase vectors: It enables us to directly study the relation between the representation of a phrase with its components. Finally, we showed by an example that phrase vectors can be integrated into down stream models in the same way word vectors do, and observed significant improvement compared to randomly initialized vectors. We also explored the possibility of extending beyond n-grams and directly getting representations for more general language features like POS-tags and synsets (see Appendix A).

These results collectively suggest that generalizing the feature set for natural language inputs and using co-occurrence statistics methods to obtain representation for them will generate useful embeddings that could provide insight into the compositional structure of language and benefit down stream NLP tasks.

5 Future Work

In terms of obtaining n-gram embeddings, we could improve the tokenization step to better account for phrases that contain stop words because these are not linguistically interesting phrases. Limited by computation power and precise evaluation metric, we are not able to fine tune some parameters for tokenization heuristics. Improving this could boost the performance of these vectors.

In terms of generalizing the scope of co-occurrence embedding, we briefly explored the possibility of obtaining synset vectors by performing word sense disambiguation on unlabeled corpus and count co-occurrence (see Appendix A). We were not able to get much quantitative results because we could only annotate a small corpus with limited accuracy. If we could overcome the computation bottleneck, we may be able to let a model directly learn on concepts. This will be very exciting.

6 Acknowledgements

We want to thank our mentor Danqi Chen for her guidance and advice for our project. We want to thank Prof. Manning, Dr. Socher and all 224N teaching staff for delivering a course about cutting edge technologies in natural language processing. Thank you.

References

[1] Yoshua Bengio, Rjean Ducharme, Pascal Vincent, Christian Jauvin, Jaz Kandola, Thomas Hofmann, Tomaso Poggio, and John Shawe-Taylor. A Neural Probabilistic Language Model. *Journal of Machine Learning Research*, 3:1137–1155, 2003.

[2] Steven Bird, Ewan Klein, and Edward Loper. *Natural Language Processing with Python*. O'Reilly Media, 2009.

[3] Danqi Chen, Richard Socher, Christopher D Manning, and Andrew Y Ng. Learning New Facts From Knowledge Bases With Neural Tensor Networks and Semantic Word Vectors. *International Conference on Learning Representations*, pages 1–4, 2013.

[4] Manaal Faruqui and Chris Dyer. Community evaluation and exchange of word vectors at wordvectors.org. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, Baltimore, USA, June 2014. Association for Computational Linguistics.

[5] Christiane Fellbaum. *WordNet: An Electronic Lexical Database*. Bradford Books, 1998.

[6] Mohit Iyyer, Varun Manjunatha, Jordan L Boyd-Graber, and Hal Daumé III. Deep unordered composition rivals syntactic methods for text classification. In *ACL (1)*, pages 1681–1691, 2015.

[7] Quoc Le and Tomas Mikolov. Distributed Representations of Sentences and Documents.

[8] Michael Lesk. Automatic sense disambiguation using machine readable dictionaries: How to tell a pine cone from an ice cream cone. In *Proceedings of the 5th Annual International Conference on Systems Documentation*, SIGDOC '86, pages 24–26, New York, NY, USA, 1986. ACM.

[9] Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60, 2014.

[10] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient Estimation of Word Representations in Vector Space.

[11] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed Representations of Words and Phrases and their Compositionality. 2013.

[12] Tomas Mikolov, Wen-Tau Yih, and Geoffrey Zweig. Linguistic Regularities in Continuous Space Word Representations.

[13] Jeffrey Pennington, Richard Socher, and Christopher D Manning. GloVe: Global Vectors for Word Representation. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1532–1543, 2014.

[14] Sascha Rothe and Hinrich Schütze. AutoExtend: Extending Word Embeddings to Embeddings for Synsets and Lexemes. *Proceedings of the ACL*, pages 1793–1803, 2015.

[15] Richard Socher, John Bauer, Christopher D Manning, and Andrew Y Ng. Parsing with Compositional Vector Grammars.

[16] Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank.

[17] Kristina Toutanova and Christopher D Manning. Enriching the knowledge sources used in a maximum entropy part-of-speech tagger. In *Proceedings of the 2000 Joint SIGDAT conference on Empirical methods in natural language processing and very large corpora: held in conjunction with the 38th Annual Meeting of the Association for Computational Linguistics-Volume 13*, pages 63–70. Association for Computational Linguistics, 2000.

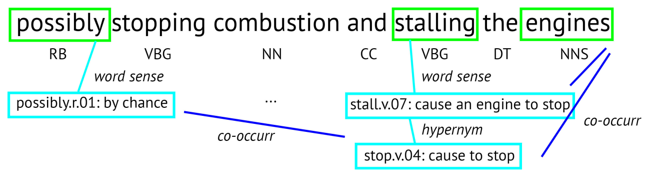

Figure 3: Extending Representation to Synsets

A given sentence is enriched by tagging POS-tags and finding synsets of component words. Light blue boxes are synsets and hypernym synsets of words in a sentence. Blue lines show some counted co-occurrence. Here the co-occurrence of "engines" with "stalling", "stall.v.07", and "stop.v.04" are all incremented.

A Extending Distributed Representation to Synsets and POS-Tags

After observing the encouraging results, especially motivated by how naturally word embedding generalizes to n-gram embedding, we decided to look at the other direction of composition relation in natural language: sub-word relations. A word encodes both semantic and syntactic information, several words may share the same meaning, or they may partially share the same meaning. For example, interest, interesting and interested share a lot of meaning, yet for most models they are treated completely differently. It would be desirable if a model can learn the meaning of word from these underlying concepts and how these concepts interact with each other. In our attempt, we used knowledge base WordNet and POS tags to enrich the features of a sentence and use the GloVe method to factorize the log co-occurrence matrix, thus obtaining the embeddings of multiple natural language features: words, n-grams, synsets and POS-tags. We call the vectors obtained combined-feature vectors.

A.1 Features

Wordnet [5] is a knowledge base organized by sets of synonyms called "synset". It contains 155287 tokens, 117659 synsets and 206941 word-sense relations. Synsets have relations between them such as "similar to", "hypernym". We try to obtain embeddings for synsets directly through co-occurrence embedding methods.

Given a sentence, we use Stanford Core NLP package [9, 17] to perform POS-tagging. We then used the NLTK package [2] to interface with Wordnet and perform word sense disambiguation with the POS tag using the built in Lesk [8] algorithm with a window size of 5. For example, a word 'stall' may appear in our corpus. It is tagged with POS-tag 'v' and synset "stall.v.07: cause an engine to stop". Synset "stall.v.07" has a hypernym "stop.v.04: cause to stop". If "car engine" is also in the context, the co-occurrence of, for example, "stall" with "car", "stop.v.04" with "engine", "stall.v.07" with "car engine" will all be incremented. Specifically, whenever a synset appears, we view those synsets that are its "hypernym", "similar to" and "also" to also appear, with a penalty coefficient 0.5 added to co-occurrence count to mitigate over-counting. Figure 3 on page 11 demonstrates an example.

A.2 Dataset

Our corpus is the first 10^8 bytes of the English Wikipedia dump on Mar. 3, 2006. They are pre-processed by the method listed at http://www.mattmahoney.net/dc/textdata.html. A feature counting run found 14,747,197 grams ($n = 1, 2, 3$), 68938 synsets, 35 tree bank POS-tags. We trimmed the vocabulary before training vectors. One test has the most frequent 60,000 n-grams (17, 912 words + 42,088 n-grams), 20,000 synsets and all 35 POS-tags, we shall refer to vectors trained on this vocabulary as **Combined**. We also trained vectors with the most frequent 60,000 words, 20,000 n-grams, 20,000 synsets and all 35 POS-tags, we shall refer to vectors trained on this vocabulary as **Combined-S** where "S" stands for "suppressing n-grams".

We are not able to train these vectors on bigger corpus mainly because of limitations of time and computing resources.

A.3 Co-occurrence factorization

The counted co-occurrence matrix is then factorized using GloVe algorithm attempting to approximate the co-occurrence X_{ij} of signals i and j with $v_i \cdot v_j \sim \log(X_{ij} + 1)$.

We used most hyper-parameters suggested by [13]. Specifically, we were used symmetric window with window size 15 and applied a $\frac{1}{n}$ penalty for separated co-occurrence where n is the number of tokens in between. These counted co-occurrence matrix is then fed into GloVe for factorization. We thus obtain vectors with 200 dimensions.

A.4 Evaluation

Because we were not able to train the combined-feature vectors on a billion token level dataset, its word-level structure will not be comparable to GloVe, Skip-Gram or our own gram vectors. At the same time, there aren't any very effective metric for synset embedding. We will present statistics on word analogy and show nearest neighbors using the 200 dimensional combined-feature vectors.

A.4.1 Word analogy

We again evaluate the obtained vector representations on word analogy task. See the second section in Table 4 on page 13. The combined vectors performed very well on its vocabulary set. But because of the increased feature set size, its vocabulary only covered around half of all questions.

The other two instances of 50 dimensional **Combined** and **Combined-S** vectors shows the decrease in accuracy when we try to increase the vocabulary size. As we increase the number of words in the vocabulary performance on word analogy degrades quickly. This is very likely because these added words do not appear frequent enough in the small corpus.

A.4.2 Nearest Neighbor

We also computed the nearest neighbor of some words in our vocabulary. They are showed in Table 5. We can see that the representation of word "acids" is be very similar to its lemma "acid", the synset of "acid" and phase "amino acids". We see that using co-occurrence methods, we can directly relate words with abstract concepts in the embedding space. These clustering structures would allow NLP models to better extract semantic information from text input.

A.5 Conclusion and Future Work (for Synset Vectors)

Our work on finding representation for synsets and concepts is limited by corpus size and annotation speed. Yet from the limited results we can still see the possibilities of seeing multiple signals from a piece of language and infer meaning from it. If we could improve annotation speed and the accuracy of word sense disambiguation, we could run this algorithm on a larger corpus and obtain high quality representation for synsets and other features. Then in a downstream task, an out of vocabulary word no longer needs to be treated as "<unk>". An application can look up some lexical database and see that "amino acids" has representation for its synset, and use that representation to infer meaning of input. If a word can be viewed as a combination of multiple components, the number of words that can be represented grows exponentially with respect to possible components. This would allow models to efficiently generalize to a much larger vocabulary.

B Effectiveness of co-occurrence factorization

Readers may question the validity of using co-occurrence factorization to obtain reliable representation of natural language signals. We shall present the results of our experiment on raw co-occurrence count to explain our motivation.

Consider the objective of GloVe when it is looking for the best word "d" for the analogy task $a, b \sim c, d$. It is maximizing over d

$$\max_d (v_a - v_b) \cdot (v_c - v_d)$$

Table 4: Word Analogy Results for Synset Vectors and RATIO algorithm

Corpus	Method	Dim	vocab	coverage	sem 1	sem 10	syn 1	syn 10	tot 1	tot 10
17M	RATIO	-	71,290	91.21	**27.99**	**74.42**	0.76	6.94	12.09	35.01
17M	GloVe*	50	71,290	91.21	27.37	52.74	**19.98**	**40.75**	**23.05**	**45.74**
17M	**Combined**	50	80,035	47.31	15.17	52.30	10.64	36.43	11.80	40.49
17M	**Combined-S**	50	100,035	86.26	9.19	32.74	7.61	25.72	8.25	28.56
17M	**Combined**	200	80,035	47.31	**22.10**	**78.96**	**12.47**	**56.84**	**14.94**	**62.50**

Models in bold face are trained by our approach. RATIO is the purely co-occurrence count algorithm discussed in Appendix B. **Combined** assigns vector representations to words, n-grams, pos-tags and synsets. **Combined-S** increases the proportion of words and decrease the proportion of grams compared to the vocabulary of **Combined**. For details see Appendix A.

acids	central bank
synset: ... water-soluble compounds having a sour taste and ...	ecb *(European Central Bank)*
amino	bank
bigram: amino acids	*synset*: a building in which the business of banking is transacted
acid	monetary

Table 5: Nearest Neighbor for Combined-Feature Vectors

If we neglect normalization, smoothing, etc. and plug in the training objective of GloVe: $v_i \cdot v_j \sim \log(X_{ij})$, we get a effective word analogy objective

$$\max_d \frac{X_{ab}X_{cd}}{X_{ad}X_{bc}}$$

call it RATIO. We use the objective of RATIO with co-occurrence statistics of 17M wiki dataset, and got a 74% top 10 accuracy on semantic questions compared to 52% of GloVe trained on the same dataset. See the first section of Table 4. This results allow us to hypothesize that the good performance of GloVe vectors on capturing semantic information *comes* from the information contained in co-occurrence count.

With the success of existing researches and our own toy experiment, we are justified to expect that by extending co-occurrence embedding to a larger scope of features, we may enable neural models to better capture the patterns of natural language.