A Study of Analogical Relationships in N-gram Embedding Models

Kun Chen[1], Hao Wang[2] and Yves Lepage[3]

Graduate School of Information, Production and Systems
Waseda University
2-7 Hibikino, Wakamatsu-ku, Kitakyushu-shi, 808-0135, Fukuoka-ken, Japan
{chenkun94@akane.,oko_ips@ruri.,yves.lepage@}waseda.jp

Abstract. This paper presents a study about analogical relationships in n-gram embedding models. We present a pipeline to produce semantic analogies between n-grams of the same length from a given corpus. We train n-gram embedding models from unigrams to 4-grams and report some interesting phenomena in these embedding models. We make an experiment of testing analogical examples on n-gram embedding models. The results show that analogical relationships exist in n-gram embedding models and the accuracy decreases when n increases.

Keywords: Analogical relationship, N-grams, Embedding models

1 Introduction

The distributed representation of concepts was proposed by Hinton in 1986 [11]. It consists in mapping each word in a language to a fixed vector. All the vectors together construct a vector space and each word is a vector in the space. In such a space, we can measure the similarity between words by computing their distance in the space. One way to generate distributed representations of words is the neural network. Bengio [1] proposed a method to train neural probabilistic language model with word vectors. Mikolov [9] simplified the training method and proposed Word2Vec which is a fast way to train on the large corpus.

Word vectors have been used in many NLP tasks such as sentiment classification [12], entity recognition [3], machine translation [8], [13]. The reason for it to be widely used is that it can capture syntactic and semantic word relationships [7] between two words which frequently appear in similar contexts. In that case, their vectors will be similar. For example, the word 'red' and the word 'blue' are used to describe colors, so that their contexts are similar most of the time. After training, their vectors will be similar and they will map onto two closed points in the word vector space.

Analogies are a general relationship between four objects: A, B, C and D. An analogy $A : B :: C : D$ states that "A is to B as C is to D". Analogical equations are the following problems: if three strings A, B and C are given, how to coin the fourth string? Lepage proposed an algorithm to solve proportional analogy on words [4]. Here, in word vector space, Mikolov find that relations between words are reflected in the offsets between their vector embeddings [10]. For example, for the analogy "*man* : *woman*

:: *king* : *queen*", in which the male/female opposition exists, we can coin the fourth word "*queen*" by computing the vector of ($\overrightarrow{woman}$-$\overrightarrow{man}$+$\overrightarrow{king}$) and look for the word closest to it.

Analogy can be used for measuring the quality of embedding models. Mikolov [10] created an analogy question set for this purpose. Some equations from this set are:

young : *younger* :: *big* : *bigger* (syntax analogy)

Beijing : *China* :: *Rome* : *Italy* (semantic analogy)

The purpose of this paper is to inquire whether the analogical relationship can also be extracted from n-gram embedding models. To achieve this goal, we divide our study into three parts. First, we generate a lot of semantic analogies between n-grams from a given corpus by using features like frequency, POS tagging and word similarity. Second, we learn n-gram embedding models on corpora of different sizes by using Word2Vec. Finally, we test the analogies in these embedding models. We do the study from bigrams to 4-grams. Experiment results show that the accuracy of n-gram analogies decreases when n increases and using the larger corpus to train embedding models leads to the better value of accuracy.

This paper is organized as follows. Section 2 describes the pipeline for the production of examples of analogies between n-grams. Section 3 describes the training process for n-gram embedding models and presents some interesting phenomena. Section 4 shows the experiment results and reports the accuracy of analogies between n-grams in our models.

2 Pipeline for The Production of Examples of Analogies between N-grams

Examples of analogies can be divided into analogies of form and analogies of meaning. Table 1 shows the difference between these two types of analogies.

Table 1. Different types of analogies

Type	Analogy
Just Form	*to give* : *give up* :: *to close* : *close up*
Just Meaning	*United States* : *Donald Trump* :: *United Kingdom* : *Theresa May*
Form and Meaning	*last week* : *next week* :: *last year* : *next year*

As it is hard to find examples of analogies which just satisfy meaning but no form, our goal is to generate analogies between n-grams which are valid on the levels of both form and meaning.

To make generated n-gram analogies more commonly, we choose a large parts of Wikipedia corpus as the original corpus. Its size is 100 million words. The production pipeline as following steps:

1. Convert unigram corpus into n-gram corpus.
2. Compute n-gram frequencies. Here we select the 30,000 most frequent n-grams.
3. Add POS tagging information to each n-gram and classify the n-grams by their POS tagging types.
4. For each POS tagging type, generate examples of analogies.
5. Use word similarity to filter meaningless examples.
6. Clean up the data manually.

Step details:

When generating examples of analogies between n-grams, most of them are satisfied on the level of surface form but not on the level of meaning, To remove these meaningless examples, we filter them by using the two following features:

The first feature is *POS tagging information*. In an example of analogy between n-grams, if the POS tagging types of the four elements are different, the chances that it makes no sense are very high. Hence at steps 3 and 4, we classify the n-grams into different POS tagging types and generate only the examples of analogies which share the same POS tagging type. In this way, we avoid the generation of a large number of meaningless analogies.

The second feature is *word similarity*. After step 4, many n-gram analogies have the similar form and same POS tagging types, but there are still some meaningless analogies, like "*make up* : *make out* :: *turn up* : *turn out*". To filter them out, we use word vectors to measure the similarity between words in the n-grams. The procedure is as follow:

For each part of words in the n-grams, we calculate their similarity. If their similarity is less than a given threshold, we filter out this analogy.

For instance, we set the threshold to 0.3 for bigram POS tagging type 'VB RP'. To justify whether an analogy like "*make up* : *make out* :: *turn up* : *turn out*" is a meaningless analogy, we calculate the similarity between '*make*' and '*turn*', and the similarity between '*up*' and '*out*' separately. We save this analogy if both values are greater than the threshold, and we filter it out if not. In this example, the similarity of '*make*' and '*turn*' being 0.25, i.e., less than the threshold, it is filtered out.

For different POS tagging types, we use different thresholds for filtering, and all the thresholds are learned from a training set of analogies.

Following this pipeline, we generate n-gram analogies from bigrams to 4-grams. Table 2 gives some examples.

Table 2. Numbers and examples of analogies between N-grams for different values of N

N	Examples	Numbers
2	*this month* : *this week* :: *next month* : *next week*	6,848
3	*with his wife* : *with her husband* :: *of his wife* : *of her husband*	5,328
4	*north of the river* : *south of the river* :: *north of the village* : *south of the village*	4,716

3 Training N-gram Embedding Models

In Sect.2, we introduced a pipeline for the production of analogies between n-grams. As our goal is to study analogical relationships between n-grams in embedding models, we create n-gram embedding models.

For that purpose, we first convert the original corpus into a corpus of n-grams. We then train n-gram embedding models from that corpus directly.

In our experiments, we use 100 million words from the Wikipedia corpus as training corpus, and use the *CBOW model* to train our vector space models from n-grams. we set parameters *Dimensions* to 200, *Min Count* to 5 and *Window Size* to 7, here the *Window Size* is for n-grams rather than words .

Table 3 shows the statistics on the corpus and models of n-grams. We find that when n increases, the average frequency of n-gram continues to decrease due to the fact that the number of n-gram types increases. The vocabulary size of the n-gram embedding model increases from unigrams to trigrams and decreases from trigrams to 4-grams due to the effect of the parameter *Min Count* which was set to 5 and to the fact that the frequencies of most 4-grams are less than 5.

Table 3. Statistics on the corpus and models of n-grams

N	N-grams	Average frequency of n-gram	Vocabulary size of models
1	694,463	143.99	184,904
2	13,891,815	7.20	1,437,762
3	41,050,454	2.43	1,523,489
4	61,643,061	1.62	884,977

Table 4 shows a few examples of results for the similarity task on our embedding models. The similarity task consists in, given an n-gram A, to return the most similar n-gram B in the embedding model, B is computed as the $\underset{B \in V}{\text{argmax}} \cos(A, B)$. We choose the top three n-grams as results. On the whole, our n-gram embedding models seem to perform well and can capture the similarity relationships between n-grams. Due to the vocabulary size of embedding model increases from 2-gram to 4-gram, the vector space is getting crowded, so the similarity also increases from 2-gram to 4-gram.

In order to further enquire the contents of the n-gram embedding models, we make a visualization of the spaces. We first use t-SNE model [6] to compress 200 dimensions into 2 dimensions, then plot the n-gram vectors by classifying them according to their POS tagging types. Because there are 36 POS tagging types in Penn Treebank for unigrams, the number of POS tagging types for n-grams is potential 36^n. To reduce the number of POS tagging types, we merge some of them. Practically we merge all tags for nouns (resp. for verbs, adjectives and adverbs) into one single tag, 'NN' (resp. 'VB', 'JJ' and 'RB'). Finally, we plot most frequent pos tagging types for each n-gram.

The visualization Figure 1 allows us to make several observations.

Firstly, the clarity of the boundaries between different POS tagging types is getting blurred when going from unigrams to 4-grams. The boundaries between the different

Table 4. Examples of results for similarity (cosine value) task with N-gram embedding models

Input N-gram	Returned N-gram
this is	(this was, 0.6380)
	(this has, 0.5656)
	(that is, 0.5478)
this is a	(this is an, 0.8373)
	(here is a, 0.8350)
	(that is a, 0.7663)
this is not a	(it is necessary to, 0.9742)
	(but this is not, 0.9740)
	(that is does not, 0.9735)
machine learning	(data mining, 0.8514)
	(computer vision, 0.8264)
	(data analysis, 0.8132)
natural language processing	(in software engineering, 0.9436)
	(of natural language, 0.9406)
	(and artificial intelligence, 0.9401)

types are relatively clear for unigrams and bigrams. However, it is difficult to distinguish boundaries for trigrams, and almost impossible for 4-grams.

In order to measure the clarity of the cluster boundaries from 1-grams to 4-grams, we calculate the values of the Davies-Bouldin index (DBI) [2].

First, we denote each n-gram as a point x_i and each POS tagging type of n-grams as a cluster $C_k = \{x_1, x_2..., x_{|C_k|}\}$. The clustering result is a set of clusters $C = \{C_1, C_2, ..., C_k\}$. The computation of DBI necessitates the definitions of the two following distances.

The internal distance of a cluster:

$$avg(C) = \frac{2}{|C|(|C|-1)} \sum_{1 \le i < j \le |C|} dist(x_i, x_j) \tag{1}$$

The distance between two clusters:

$$d_{cen}(C_i, C_j) = dist(\mu_i, \mu_j) \tag{2}$$

In the above formula, $dist(\cdot, \cdot)$ is the distance between two points, μ is the center point of a cluster: $\mu = \frac{1}{|C|} \sum_{1 \le i \le |C|} x_i$.

The $avg(C)$ distance is the average distance between all the points in a cluster C. It measures the degree of dispersion of a cluster. The higher the $avg(C)$ value, the greater the degree of dispersion. The $d_{cen}(C_i, C_j)$ distance is the distance between cluster centers μ_i and μ_j.

$$DBI = \frac{1}{k} \sum_{i=1}^{k} \max_{j \ne i} \left(\frac{avg(C_i) + avg(C_j)}{d_{cen}(\mu i, \mu j)} \right) \tag{3}$$

The Davies-Bouldin index is defined as the ratio of the sum of all internal distances of all clusters to the distance between each cluster pairs. The smaller the DBI value, the clearer the cluster boundaries. We calculate the DBI for the above visualizations of

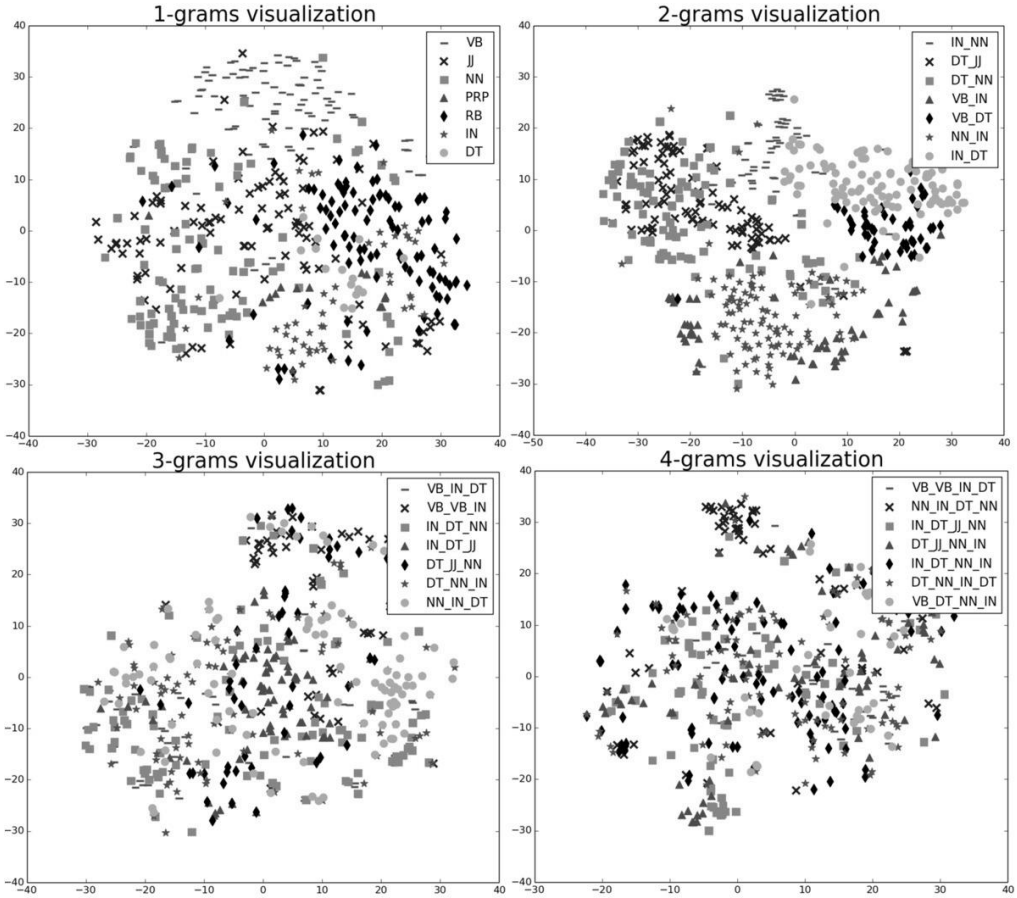

Fig. 1. Visualization of embedding models from 1-gram to 4-gram

Table 5. The value of *DBI* for visualization of different N-grams

N-grams	1	2	3	4
DBI	47.61	48.73	73.63	77.30

n-grams. Table 5 shows that the clarity of the boundaries between different POS tagging types is getting blurred when going from unigrams to 4-grams.

Secondly, the POS tagging types of two n-grams share the same subpart, their vectors are found in a close region. Many examples confirm this impression. For instance:

The bigrams with POS tagging types 'DT JJ' and 'DT NN' are closer than to other types because they share 'DT'.

The trigrams with POS tagging types 'IN DT JJ' and 'DT JJ NN' are closer than to other types because they share 'DT JJ'.

To measure this phenomenon, we calculate the distances between each cluster pairs. Table 6 shows the distances between each cluster pairs in trigrams. The boldface values show the smallest values on each line. It shows that when trigram POS tagging types of two clusters share the same subpart, their distance is smaller than to other types.

Table 6. Distances between trigram POS tagging types

	VB IN DT	IN DT NN	VB IN IN	IN DT JJ	DT JJ NN	DT NN IN	NN IN DT
VB IN DT		0.403	**0.274**	0.455	0.388	0.321	0.403
IN DT NN	0.403		0.375	0.420	0.336	**0.270**	0.455
VB VB IN	**0.274**	0.375		0.518	0.415	0.390	0.456
IN DT JJ	0.455	0.420	0.518		**0.367**	0.470	0.559
DT JJ NN	0.386	0.336	0.415	0.367		**0.267**	0.401
DT NN IN	0.321	0.270	0.390	0.470	**0.267**		0.314
NN IN DT	0.403	0.455	0.456	0.559	0.401	**0.314**	

4 Accuracy of Analogies between N-grams

In Sec.2 and Sec.3, we introduced the pipeline for the production of analogies between n-grams and the training of n-gram embedding models. To verify whether analogical relationships can be found between n-grams in embedding models, we perform an experiment of testing n-gram analogies on n-gram embedding models.

Table 7 shows some settings for the experiment. We use the Wikipedia corpora and intersect corpora of different sizes to train embedding models from bigrams up to 4-grams. We use generated n-gram analogies which introduced in Sec.2 as test dataset.

Table 7. Corpora sizes and parameters for the CBOW model of Word2Vec

N-grams	2, 3, 4
Word2vec Model	CBOW model
Corpus Size	1M, 10M, 100M (M = million words)
Dimension	200
Window Size	7
Min Count	2 for 1M, 3 for 10M, 5 for 100M

We choose 3CosAdd function as the analogy test function [5]:

$$\underset{b^* \in V}{\text{argmax}}(cos(b^*, b) - cos(b^*, a) + cos(b^*, a^*)) \tag{4}$$

The interpretation of the formula is as follows: look for a word b^* which is similar to b and a^* but different from a from all the vocabulary V. Table 8 shows the results of experiments.

Table 8. Accuracy on examples of analogies in n-gram embedding models

Corpus Size	N-gram	Top N accuracy (%)			Coverage (%)
		N=1	N=10	N=100	
1 M	2	0.01	0.14	0.56	73.77
	3	0.00	0.03	0.48	40.14
	4	0.00	0.03	0.03	1.77
10 M	2	11.69	28.72	47.02	97.47
	3	0.36	1.77	5.29	93.34
	4	0.00	0.11	0.69	59.57
100 M	2	67.58	92.37	96.74	100.00
	3	35.57	58.62	74.63	100.00
	4	8.72	21.83	38.43	100.00

In table 8, 'Top N' means correct result appears in top N. From the result, we can find following results.

1. The accuracy increases with the size of the corpora used to build the embedding models increases. Using larger corpora to train embedding models leads to a better value of accuracy.

2. Another result is that the accuracy decreases as the size of the n-grams increases, but we still can find analogical relationships in these n-gram embedding models. For 4-grams, an accuracy of 8.72% can be reached for Top 1 and 38.43% for Top 100.

5 Conclusion

In this paper, we have presented the experiments in testing analogical relationships between n-grams in embedding models. We first presented a pipeline for the production

of examples of semantic analogies between n-grams starting from a given corpus. We used POS tagging information and word similarity to generate examples of analogies between n-grams on both levels of surface form and meaning. Second, we trained embedding models from unigrams up to 4-grams. We then performed similarity tests and visualization experiments on these models. Finally, we tested examples of analogies between n-grams on embedding models. The result showed that analogical relationships exist in n-gram embedding models. We believe that using larger corpus will increase the accuracy of such results.

References

1. Bengio, Y., Ducharme, R., Vincent, P., Janvin, C.: A neural probabilistic language model. vol. 3, pp. 1137–1155. JMLR.org (Mar 2003)
2. Davies, D.L., Bouldin, D.W.: A cluster separation measure. IEEE Trans. Pattern Anal. Mach. Intell. 1(2), 224–227 (Feb 1979)
3. Lample, G., Ballesteros, M., Subramanian, S., Kawakami, K., Dyer, C.: Neural architectures for named entity recognition. In: Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL 2016). pp. 260–270. Association for Computational Linguistics, San Diego, California (June 2016)
4. Lepage, Y.: Solving analogies on words: an algorithm. In: Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics (COLING-ACL 1998). pp. 728–734 (1998)
5. Levy, O., Goldberg, Y.: Linguistic regularities in sparse and explicit word representations. In: Proceedings of the Eighteenth Conference on Computational Natural Language Learning (ACL 2014). pp. 171–180. Association for Computational Linguistics, Ann Arbor, Michigan (June 2014)
6. van der Maaten, L., Hinton, G.E.: Visualizing high-dimensional data using t-sne. Journal of Machine Learning Research 9, 2579–2605 (2008)
7. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. In: Proceedings of International Conference on Learning Representations (ICLR 2013) (2013)
8. Mikolov, T., Le, Q.V., Sutskever, I.: Exploiting similarities among languages for machine translation. In: Proceedings of International Conference on Learning Representations (2013)
9. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Advances in Neural Information Processing Systems 26 (NIPS 2013). pp. 3111–3119. Curran Associates, Inc. (2013)
10. Mikolov, T., Yih, W.t., Zweig, G.: Linguistic regularities in continuous space word representations. In: Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2013). pp. 746–751. Association for Computational Linguistics, Atlanta, Georgia (June 2013)
11. Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Parallel distributed processing: Explorations in the microstructure of cognition, vol. 1. pp. 318–362. MIT Press, Cambridge, MA, USA (1986)
12. Tang, D., Wei, F., Yang, N., Zhou, M., Liu, T., Qin, B.: Learning sentiment-specific word embedding for twitter sentiment classification. In: Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL 2014). pp. 1555–1565. Association for Computational Linguistics, Baltimore, Maryland (June 2014)

13. Zhang, J., Liu, S., Li, M., Zhou, M., Zong, C.: Bilingually-constrained phrase embeddings for machine translation. In: Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL 2014). pp. 111–121. Association for Computational Linguistics, Baltimore, Maryland (June 2014)