K-Embeddings: Learning Conceptual Embeddings for Words using Context

Thuy Vu and **D. Stott Parker**
Department of Computer Science
University of California, Los Angeles
Los Angeles, CA 90095
{thuy;stott}@cs.ucla.edu

Abstract

We describe a technique for adding contextual distinctions to word embeddings by extending the usual embedding process — into two phases. The first phase resembles existing methods, but also constructs K classifications of concepts. The second phase uses these classifications in developing refined K embeddings for words, namely word K-embeddings. The technique is iterative, scalable, and can be combined with other methods (including Word2Vec) in achieving still more expressive representations.

Experimental results show consistently large performance gains on a Semantic-Syntactic Word Relationship test set for different K settings. For example, an overall gain of 20% is recorded at $K = 5$. In addition, we demonstrate that an iterative process can further tune the embeddings and gain an extra 1% ($K = 10$ in 3 iterations) on the same benchmark. The examples also show that polysemous concepts are meaningfully embedded in our K different conceptual embeddings for words.

1 Introduction

Neural-based word embeddings are vectorial representations of words in high dimensional real valued space. Success with these representations have resulted in their being considered for an increasing range of natural language processing (NLP) tasks. Recent advances in word embeddings have shown great effects that are pushing forward state-of-the-art results in NLP (Koo et al., 2008; Turian et al., 2010; Collobert et al., 2011; Yu et al., 2013; Mikolov et al., 2013a; Mikolov et al., 2013b; Mikolov et al., 2013c). Embedding learning models for words are also being adapted for tasks in other research fields (Reinanda et al., 2015; Vu and Parker, 2015). The Continuous bag of words (CBOW) and Skip-gram (Mikolov et al., 2013a) are currently considered as state-of-the-art in learning algorithms for word embeddings.

The ability of words to assume different roles (syntax) or meanings (semantics) presents a basic challenge to the notion of word embedding (Erk and Padó, 2008; Reisinger and Mooney, 2010; Huang et al., 2012; Tian et al., 2014; Neelakantan et al., 2014; Chen et al., 2015). External resources and features are introduced to address this challenge. In general, individuals with no linguistic background can generally resolve these differences without difficulty. For example, they can distinguish "bank" as referring to a riverside or a financial establishment without semantic or syntactic analysis.

Distinctions of role and meaning often follow from context. The idea of exploiting context in linguistics was introduced with a distributional hypothesis: "linguistic items with similar distributions have similar meanings" (Harris, 1954). Firth soon afterwards emphasized this in a famous quote: "a word is characterized by the company it keeps" (1957).

We propose to exploit only context information to distinguish different concepts behind words in this paper. The contribution of this paper is to note that a *two-phase word embedding* training can be helpful in adding contextual information to existing embedding methods:

- we use learned context embeddings to effi-

Proceedings of NAACL-HLT 2016, pages 1262–1267,
San Diego, California, June 12-17, 2016.

ciently cluster word contexts into K classifications of concepts, independent of the word embeddings.

- this approach can complement existing sophisticated, linguistically-based features, and can be used with word embeddings to achieve gains in performance by considering contextual distinctions for words.

- two-phase word embedding may have other applications as well, conceivably permitting some 'non-linear' refinements of linear embeddings.

In the next section we present our learning strategy for word K-embeddings, outlining how the value of K affects its power in increasing syntactic and semantic distinctions. Following this, a large-scale experiment serves to validate the idea — from several different perspectives. Finally, we offer conclusions about how adding contextual distinctions to word embeddings (with our second phase of embedding) can gain power in distinguishing among different aspects of words.

2 Learning Word K-Embeddings

The use of multiple semantic representations for a word in resolving polysemy has a significant literature (Erk and Padó, 2008; Reisinger and Mooney, 2010; Huang et al., 2012; Tian et al., 2014; Neelakantan et al., 2014; Chen et al., 2015). Strategies often focus on discrimination using syntactic and semantic information.

We investigate another direction — the extension of the word embedding process into a second phase — which allows context information to be consolidated with the embedding. Rather than annotating words with features, our technique treats context as second-order in nature, suggesting an additional representation step.

Our learning strategy for word K-embeddings is therefore done, possibly iteratively, in two phases:

1. Annotating words with concepts (defined by their contextual clusters)

2. Training embeddings using the resulting annotated text.

2.1 Concept Annotation using Context Embeddings

We propose to annotate words with concepts given by learned context embeddings, which are an underutilized output of word embedding training. Our strategy is based on the assumption that the context of a word is useful for discriminating its conceptual alternatives in polysemy. In general, our concept annotation for words is performed in two steps — clustering of context embeddings followed by annotation.

Specifically, we first employ a clustering algorithm to cluster the context embeddings. K-means is our algorithm of choice. The clustering algorithm will assign each context word to a distinct cluster. This result is then used to re-assign words in training data to their contextual cluster.

Second, we annotate words in the training data with their most common contextual cluster (of their context words). We define *context words* to mean the surrounding words of a given word. Formally, a word is annotated with a concept given by the following function:

$$\max_{c \in \mathcal{C}} \sum_{(w_i, c_i) \in W} f(c_i, c)$$

Here W is the set of context words of the current word, and $f(c_i, c_j)$ is a boolean function whose output is 1 if the input parameters are equal:

$$f(c_i, c_j) = \begin{cases} 1, & \text{if } c_i = c_j \\ 0, & \text{otherwise.} \end{cases}$$

The cluster-annotated dataset is then passed into the next training phase.

2.2 Training Word K-Embeddings

The second phase is similar to existing word embedding training systems. The number of clusters K defines the maximum number of different representations for words. Table 1 presents the statistics for different selections of K using the dataset mentioned in the **Experiments** section.

Each value K in Table 1 is shown with the total number of embeddings and vocabulary size. Words in the vocabulary can have up to K different embeddings for different annotated concepts. As K increases, the size of the vocabulary decreases —

K	total embeddings	vocabulary size	ratio
1	1,965,139	1,965,139	1.00
5	2,807,016	1,443,061	1.95
10	2,740,351	1,474,704	1.86
15	3,229,945	1,374,055	2.35
20	3,236,882	1,410,521	2.29
25	3,382,722	1,383,162	2.45
30	3,404,150	1,418,027	2.40

Table 1: Total embeddings and vocabulary size for different K for Wikipedia dataset. Words with frequency lower than 5 are filtered during pre-processing.

yet remains largely stable for different values of K greater than 1. This is explained by the count of words being scattered to different concepts, resulting in a lower word count per concept. In our setting, concept-annotated words with fewer than 5 occurrences will be discarded during training of word embeddings.

It is interesting to note that the total number of embeddings is broadly stable and less affected by K. For example, as we allow up to 10 different concepts for a word ($K = 10$), the total number of embeddings grows only slightly compared to the result for $K = 1$. The average number of embeddings for a word is 1.86 for $K = 10$. In other words, concept annotations do converge as we increase K.

2.3 Word K-Embedding Training Workflow

Figure 1 presents our proposed workflow to train context-based conceptual word K-embeddings. Our system allows each word to have at most K different embeddings, where each is a representation for a certain concept.

The input to the workflow is a large-scale text dataset. Initially, we compute context embeddings for words as presented previously. We can derive context embeddings directly from the training of almost any context-based word embeddings, where word embeddings are computed via their context words.

Subsequently, we cluster context embeddings into groups which reflect varied concepts on some semantic vector space. Each context embedding is assigned to a cluster denoting its conceptual role as a context word. Any clustering algorithm for vectors can be applied for this task.

Embeddings of annotated context words are used

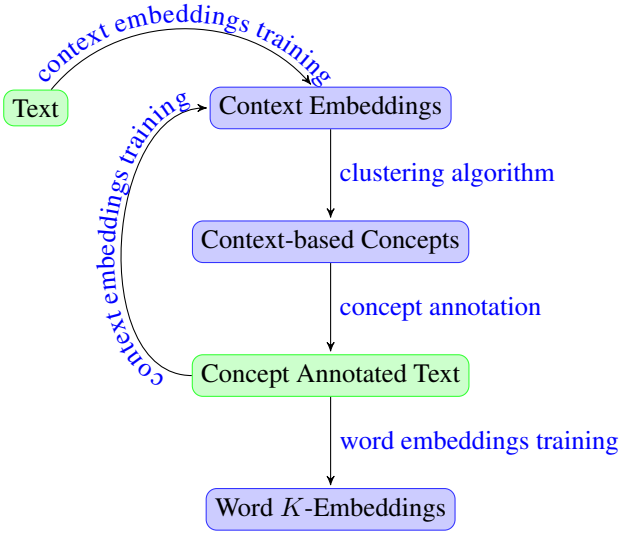

Figure 1: Training Word K-Embeddings

to compute concepts of words in a sentence. We hypothesize that the concept of a word is defined by the concept of its surrounding words. We annotate concepts for all words in the training data.

Finally, the concept-annotated training data is passed into any standard algorithm for training word embeddings for the conceptual word K-embeddings.

3 Experiments

3.1 Settings

Our training data for word embeddings is Wikipedia for English, downloaded on November 29, 2014. It consists of 4,591,457 articles, with a total of 2,015,823,886 words. The dataset is pre-processed with sentence and word tokenization. We convert text to lower-case prior to training. We consider $|W| = 5$ for the size of the context window W presented in Section 2.1.

We used the Semantic-Syntactic Word Relationship test set (Mikolov et al., 2013a) for our experimental studies. This dataset consists of 8,869 semantic and 10,675 syntactic queries. Each query is a tuple of four words (A, B, C, D) for the question "A is to B as C to what?". These queries can be either semantic or syntactic. D, to be predicted from the learned embeddings, is defined as the closest word to the vector $(A - B + C)$. We used Word2Vec for training and `scikit-learn` for clustering tasks.

We evaluate the accuracy of the prediction of D in these queries. A query is considered hit if there exists at least one correct match and all the words are in the same concept group. This is based on the assumption that if "A is to B as C is to D", either (A, B) and (C, D) OR (A, C) and (B, D) have to be in the same concept group.

3.2 Results

The embeddings learned in phases 1 and 2 can be compared, using different values for K in the K-means clustering. Word relationship performance results are shown in Table 2.

Our proposed technique in phase 2 achieves consistently high performance. For example, when $K = 5$, our absolute performance is 89% and 81% in semantic and syntactic relationship evaluations, gaining 24% and 16% from the standard CBOW model (phase 1). When $K = 25$, the performance yields the best combined result. As shown in Table 1, the total number of embeddings and vocabulary size differ by a small multiplicative factor as K increases.

In another comparison, Figure 2 plots our K-Embeddings results versus the results of a *relaxed* evaluation for CBOW, which considers the top K embeddings instead of the best. Even though our evaluation is restricted to one-best for each of the K embeddings, the overall (combined) performance for different K settings is still consistently better than the top K embeddings of CBOW. Moreover, for a specific K setting, the total number of different embeddings considered in K-Embeddings is always less than that of the top K. For example, in our peak result ($K = 25$), the total number of embeddings considered in the evaluation set is only about 76.17% of the total embeddings with the top 25 of CBOW.

In addition, we also compare the performances of K-embeddings in multiple iterations under the same K setting in Table 3. It shows that the K-embeddings are improved after certain number of iterations. Particularly, for $K = 10$, we can achieve best performance after 3 to 4 iterations, gaining roughly 1%.

Finally, it is also worth noting that the performance does not always increase linearly with the number of embeddings or vocabulary size. This suggests that as we achieve better performance in K-embeddings, we should also gain more compact conceptual embeddings.

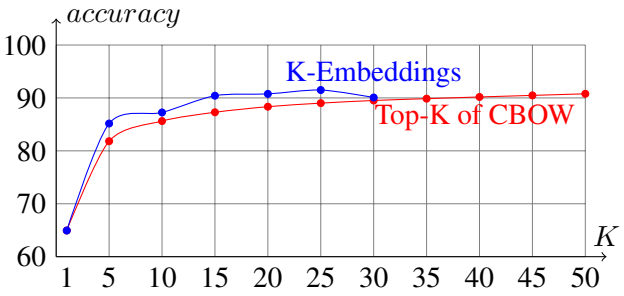

Figure 2: Word K-Embeddings and Top-K of CBOW accuracy comparison

Type	iter_1	iter_2	iter_3	iter_4	iter_5
Semantic	88.8	89.6	**90.3**	90.0	89.0
Syntactic	85.9	84.8	**86.7**	**86.7**	85.8
Combined	87.3	87.0	**88.3**	88.2	87.3

Table 3: Performance of $K = 10$ in five iterations

3.3 Word Expressivity Analysis

Expressivity of word groups for "mercury" and "fan" are studied in Table 4. The first two rows shows most related words of "mercury" and "fan" without concepts annotation (baseline). The following rows present our K-embeddings result. This table illustrates the differences that arise in multiple representations of a word, and shows semantic distinctions among these representations.

For example, different representations for the word "mercury" indeed represent a spectrum of aspects for the word, ranging from related-cosmos, related chemical element, automobile, or even to music. The same can be seen for "fan" — where we find concepts related to fan as a follower/supporter, fan as in machinery, or Fan as a common Chinese surname. Indeed, we can find many different conceptual readings of these words. These not only reflect different polysemous meanings, but also their conceptual aspects in the real world. Observe that most related words are grouped into distinct concept groups, and thus yield strong semantic distinctions. The result firmly suggests that context embeddings, like word embeddings, can capture linguistic regularities efficiently.

Analogy Type	Total	CBOW	$K=5$	$K=10$	$K=15$	$K=20$	$K=25$	$K=30$
capital-common-countries	506	85.18	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**
capital-world	4,524	78.89	96.60	97.24	99.12	99.18	**99.29**	99.27
currency	866	20.01	36.72	31.18	40.65	41.22	**42.84**	44.80
city-in-state	2,467	44.75	90.76	89.26	95.42	97.16	97.28	**97.61**
family	506	85.38	96.25	99.01	97.23	98.42	97.83	**99.21**
total semantic evaluation	8,869	64.67	89.30	88.83	92.32	92.96	93.18	**93.53**
gram1-adjective-to-adverb	992	19.76	51.41	59.98	65.73	68.95	**70.06**	61.90
gram2-opposite	812	26.72	42.12	48.52	62.81	62.19	**68.84**	56.03
gram3-comparative	1,332	87.99	97.30	97.82	**99.25**	99.17	99.02	**99.25**
gram4-superlative	1,122	52.65	71.93	74.15	**81.02**	78.88	78.70	75.22
gram5-present-participle	1,056	64.96	87.31	91.57	93.47	92.52	**96.78**	94.03
gram6-nationality-adjective	1,599	90.87	93.62	94.81	93.87	94.68	**95.37**	94.93
gram7-past-tense	1,560	65.51	66.28	94.42	94.49	95.45	**96.41**	93.72
gram8-plural	1,332	77.40	93.92	95.42	**97.90**	96.55	95.80	96.92
gram9-plural-verbs	870	66.78	83.33	94.60	94.71	95.63	**95.75**	93.22
total syntactic evaluation	10,675	65.17	81.72	85.94	88.83	88.93	**90.08**	87.21
total combined evaluation	19,544	64.94	85.16	87.25	90.42	90.76	**91.49**	90.08

Table 2: K-embeddings performance

#	Word	Most Similar Words
0	mercury	cadmium, barium, centaur, jupiter, venus
	fan	fans, fanbase, fan-base, supporter, fandom
1	mercury_1	vanadium_1, iron_1, sulfur_1, polonium_1
	mercury_3	tribune_3, dragon_3, curlew_3, keith_3, stanley_3
	mercury_5	ammonia_5, magnesium_5, sulfur_5, mercury_9
	mercury_9	mercury_5, mercury_2, neptune_9, titan_9
1	fan_1	inlet_1, crinoids_1, sect_1, wedge_1, beach_1
	fan_4	supporter_4, likes_4, legend_4, bust_4, member_4
	fan_5	fan_9, fan_0, fans_5, fandom_5, fanbase_5, gamer_5
	fan_8	xiang_8, yong_8, xin_8, yang_8, cui_8, guo_8
3	mercury_1	polaris_1, mercury_3, mercury_6, cadmium_1
	mercury_4	chrysler_4, sheedy_4, mohammad_4, gott_4
	mercury_6	arsenic_6, lithium_6, oxygen_6, methane_6, dust_6
	mercury_7	cadmium_7, nickel_7, pollutants_7, impurities_7
	mercury_8	rubidium_8, xenon_8, selenium_8, cadmium_8
3	fan_2	yong_2, ye_2, ching_2, hao_2, yi_2, chang_2, guo_2
	fan_4	member_4, parody_4, protg_4, supporter_4
	fan_6	fanbase_6, buzz_6, fans_3, fandom_6, video_6
	fan_7	imprints_7, gnatcatchers_7, minuta_7, flat_7
	fan_8	impeller_8, inlet_8, spinner_8, spring_8, hot_8
5	mercury_2	titanium_2, jupiter_2, sapphire_2, saturn_2
	mercury_3	sodium_3, helium_3, oxygen_3, hydrogen_3
	mercury_6	blue_6, leopards_6, lotus_6, unilever_6, copper_6
	mercury_7	arsenic_7, sulfur_7, radioactivity_7, lithium_7
	mercury_9	chlorine_9, strontium_9, ammonia_9, arsenic_9
5	fan_2	buzz_2, fanbase_2, gamer_2, loudest_1, fans_1
	fan_3	blower_3, ducts_3, cooler_3, compressor_3
	fan_5	zang_5, huang_5, yan_5, dun_5, zhang_5, kao_5
	fan_6	youngster_6, participant_6, mobster_6
	fan_7	fanbase_7, buzz_7, youtube_7, blogging_7
	fan_8	supporter_8, fandom_8, enthusiast_8, parody_8

Table 4: Word Expressivity Analysis

4 Conclusion

In this paper, we have presented a technique for adding contextual distinctions to word embeddings with a second phase of embedding. This contextual information gains power in distinguishing among different aspects of words. Experimental results with embedding of the English variant of Wikipedia (over 2 billion words) shows significant improvements in both semantic- and syntactic- based word embedding performance. The result also presents a wide range of interesting concepts of words in expressivity analysis.

These results strongly support the idea of using context embeddings to exploit context information for problems in NLP. As we highlighted earlier, context embeddings are underutilized, even though word embeddings have been extensively exploited in multiple applications.

Furthermore, the contextual approach can complement existing sophisticated, linguistically-based features, and can be combined with other learning methods for embedding. These results are encouraging; they suggest that useful extensions of current methods are possible with two-phase embeddings.

References

Tao Chen, Ruifeng Xu, Yulan He, and Xuan Wang. 2015. Improving distributed representation of word sense via wordnet gloss composition and context clustering. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 15–20, Beijing, China, July. Association for Computational Linguistics.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *J. Mach. Learn. Res.*, 12:2493–2537, November.

Katrin Erk and Sebastian Padó. 2008. A structured vector space model for word meaning in context. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '08, pages 897–906, Stroudsburg, PA, USA. Association for Computational Linguistics.

J. Firth. 1957. *A Synopsis of Linguistic Theory 1930-1955*. Studies in Linguistic Analysis, Philological. Longman.

Zellig Harris. 1954. Distributional structure. *Word*, 10(23):146–162.

Eric H. Huang, Richard Socher, Christopher D. Manning, and Andrew Y. Ng. 2012. Improving word representations via global context and multiple word prototypes. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers - Volume 1*, ACL '12, pages 873–882, Stroudsburg, PA, USA. Association for Computational Linguistics.

Terry Koo, Xavier Carreras, and Michael Collins. 2008. Simple semi-supervised dependency parsing. In *Proceedings of ACL-08: HLT*, pages 595–603. Association for Computational Linguistics.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In C.J.C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc.

Tomas Mikolov, Wen tau Yih, and Geoffrey Zweig. 2013c. 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)*. Association for Computational Linguistics, May.

Arvind Neelakantan, Jeevan Shankar, Alexandre Passos, and Andrew McCallum. 2014. Efficient non-parametric estimation of multiple embeddings per word in vector space. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1059–1069, Doha, Qatar, October. Association for Computational Linguistics.

Lin Qiu, Yong Cao, Zaiqing Nie, Yong Yu, and Yong Rui. 2014. Learning word representation considering proximity and ambiguity.

Ridho Reinanda, Edgar Meij, and Maarten de Rijke. 2015. Mining, ranking and recommending entity aspects. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '15, pages 263–272, New York, NY, USA. ACM.

Joseph Reisinger and Raymond J. Mooney. 2010. Multi-prototype vector-space models of word meaning. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, HLT '10, pages 109–117, Stroudsburg, PA, USA. Association for Computational Linguistics.

Fei Tian, Hanjun Dai, Jiang Bian, Bin Gao, Rui Zhang, Enhong Chen, and Tie-Yan Liu. 2014. A probabilistic model for learning multi-prototype word embeddings. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 151–160, Dublin, Ireland, August. Dublin City University and Association for Computational Linguistics.

Joseph Turian, Lev-Arie Ratinov, and Yoshua Bengio. 2010. Word representations: A simple and general method for semi-supervised learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 384–394. Association for Computational Linguistics.

Thuy Vu and D. Stott Parker. 2015. Node embeddings in social network analysis. In *Proceedings of the 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2015*, ASONAM '15, pages 326–329, New York, NY, USA. ACM.

Mo Yu, Tiejun Zhao, Daxiang Dong, Hao Tian, and Dianhai Yu. 2013. Compound embedding features for semi-supervised learning. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 563–568. Association for Computational Linguistics.