Learning Word Representation Considering Proximity and Ambiguity

Lin Qiu [†‡] * and **Yong Cao** [†] and **Zaiqing Nie** [†] and **Yong Rui** [†]
[†] Microsoft Research, {yongc, znie, yongrui}@microsoft.com
[‡] Shanghai Jiao Tong University, lqiu@apex.sjtu.edu.cn

Abstract

Distributed representations of words (aka word embedding) have proven helpful in solving natural language processing (NLP) tasks. Training distributed representations of words with neural networks has lately been a major focus of researchers in the field. Recent work on word embedding, the Continuous Bag-of-Words (CBOW) model and the Continuous Skip-gram (Skip-gram) model, have produced particularly impressive results, significantly speeding up the training process to enable word representation learning from large-scale data. However, both CBOW and Skip-gram do not pay enough attention to word proximity in terms of model or word ambiguity in terms of linguistics. In this paper, we propose Proximity-Ambiguity Sensitive (PAS) models (i.e. PAS CBOW and PAS Skip-gram) to produce high quality distributed representations of words considering both word proximity and ambiguity. From the model perspective, we introduce proximity weights as parameters to be learned in PAS CBOW and used in PAS Skip-gram. By better modeling word proximity, we reveal the strength of pooling-structured neural networks in word representation learning. The proximity-sensitive pooling layer can also be applied to other neural network applications that employ pooling layers. From the linguistics perspective, we train multiple representation vectors per word. Each representation vector corresponds to a particular group of POS tags of the word. By using PAS models, we achieved a 16.9% increase in accuracy over state-of-the-art models.

Introduction

High-quality distributed representations of words have proven helpful in many learning algorithms for speech recognition, image annotation, machine translation and other NLP tasks (Schwenk and Gauvain 2004; Schwenk, Dchelotte, and Gauvain 2006; Schwenk 2007; Weston, Bengio, and Usunier 2011; Mnih and Hinton 2007; 2008; Collobert and Weston 2008; Collobert et al. 2011). Traditionally, a word is represented by a *one-hot-spot* vector. The vector size equals the vocabulary size. The element at the word index is "1" while the other elements are "0"s. However, the *one-hot-spot* representation has two weaknesses: the vocabulary size keeps increasing with the growth of big data which leads to the curse of dimensionality (Bengio et al. 2003), and the *one-hot-spot* representation captures no syntactic or semantic regularities of words because the distances between any two words in the vector space are the same.

The distributed representation of words has garnered significant attention in the recent past. Instead of a *one-hot-spot* vector, a word is represented by a real-valued vector with a much smaller size (normally by several hundreds). Such distributed representation does not face the-curse-of-dimensionality problem since the growth of the distributed vector size is logarithmic compared to the vocabulary's growth. Moreover, the syntactic and semantic regularities of words can be encoded in the distributed vector space: the Euclidean distance between two words in the vector space represents the syntactic or semantic similarity between them. Mikolov et al. (Mikolov et al. 2013a) find that distributed word representation can preserve not only syntactic and semantic regularities, but also linear regularities. For example, $vector(\text{“}king\text{”}) - vector(\text{“}man\text{”}) + vector(\text{“}woman\text{”})$ results in a vector that is closest to $vector(\text{“}queen\text{”})$. They design a test set to measure the regularities preserved in the distributed word representation. They also carry out two neural network models for representation learning: CBOW and Skip-gram. CBOW uses a word's context words in a surrounding window to predict the word, while Skip-gram uses only one context word for prediction. Specifically, a sum pooling layer is employed in CBOW to speed up its training process. This makes it possible to train CBOW on very large-scale data which can hardly be handled with other neural network bag-of-words models (Bengio et al. 2003). Theoretically, CBOW should be superior since more context words are involved. However, Skip-gram achieves the best accuracy on their test set over all existing word representation learning models.

There is a significant performance gap between CBOW and Skip-gram. We find this comes from the proximity modeling of the context words in CBOW. CBOW is actually a classifier. The output class label is the target word while the input features are the context words which are located in a window around the target word. In CBOW, the representation vectors of the context words are fed to the sum pooling layer. The sum pooling layer treats each context word

*This work was done when the first author was on an internship with Microsoft Research.

equally by adding up the representation vectors of the context words. That is, switching any two context words will not change the pooling layer output. Therefore, the order information (or the proximity to the target word) of the context words is completely removed in CBOW. The ignorance of proximity results in poorly positioned word representations. Mikolov et al. try to perceive the context word proximity by adjusting the context window size randomly whenever a training sentence is fed to CBOW. The window size is drawn from a prior probability distribution in which the probability of selecting a certain window size drops linearly as the size becomes large. We call this strategy *dynamic window size*. *Dynamic window size* reduces the impact of proximity ignorance by choosing more small window sizes. However, *dynamic window size* is not a fundamental solution but a trade-off: using less context information to avoid negative impact. Moreover, the output vector of the sum pooling layer suffers from scale fluctuation by using *dynamic window size* since the number of input vectors changes all the time. Such scale fluctuation is eventually transmitted to the word representation vectors during error *back-propagation* (Rumelhart, Hintont, and Williams 1986). Skip-gram is relatively less sensitive to proximity since it actually captures the averaged co-occurrences of two words over the whole training set. The influence of the local context proximity is thus reduced but still exists. Also, there is no scale fluctuation issue in Skip-gram.

Besides neural network models, learning good word representations also relies on linguistics. It's common for a word to belong to multiple lexical categories. For example, the word "account" can be either a noun or a verb. It's very hard to capture the syntactic regularities of the verb "account" and the noun "account" in one representation vector simultaneously, because the vector is required to be close to a number of nouns and verbs in the vector space. Therefore, such morphosyntactic ambiguity must be considered in representation learning.

In the paper, we propose two PAS models: PAS CBOW and PAS Skip-gram for producing high quality word representations considering both word proximity and ambiguity. Since the lexical categories of a word are represented by its POS tags, we focus on the POS ambiguity, i.e. a word possibly having multiple valid POS tags. We attack the POS ambiguity problem by creating multiple representation vectors for one word. Besides creating one vector for each POS tag, we also try creating vectors for particular groups of POS tags since the occurrences of a word may hold the same meaning even when their POS tags are different. We model word proximity in PAS CBOW by introducing proximity weights. They are treated as a special network layer which is placed before the pooling layer. These weights are updated during training. By introducing proximity weights, we fix the context window size so that fluctuations in word representation vectors are removed as the scales of the projection vector items are stabilized. Although learning the proximity weights creates additional calculation cost, the total training time of PAS CBOW is still comparable to CBOW. Moreover, the proximity weight layer can also be employed in other neural network applications, that have pooling layers, to model proximity. In PAS Skip-gram, we model the word proximity by leveraging the proximity weights learnt after the training of PAS CBOW. Specifically, we achieve an accuracy increase of 16.9% with PAS CBOW and 3.7% with PAS Skip-gram.

Related Work

The distributed representation of words is carried out in (Hinton 1986; Elman 1991). Word representation is then used in learning language models. Bengio et al. (Bengio et al. 2003) propose a neural network language model (NNLM) which uses the context words in a window to predict the next word. NNLM consists of a sequential projection layer, in which the context word representation vectors are concatenated, as are classification layers. Word proximity does not need to be modeled explicitly since the context word order is already considered in concatenation. NNLM outperforms traditional N-gram models and is applied to a variety of learning tasks in speech recognition, machine translation and image annotation (Schwenk and Gauvain 2004; Schwenk, Dchelotte, and Gauvain 2006; Schwenk 2007; Weston, Bengio, and Usunier 2011). Morin et al. (Morin and Bengio 2005) propose a hierarchical architecture which significantly improves the training speed of NNLM. Mnih et al. (Mnih and Hinton 2007; 2008; Mnih and Teh 2012) further improve both model performance and training speed.

Instead of focusing on learning language models, Collobert et al. (Collobert and Weston 2008; Collobert et al. 2011) are interested in learning word representations directly. They learn word representations in a binary classification task: whether the word in the middle of a window is related to its context word in the window or not. They use the learned word representations to initialize the neural network models for other NLP tasks that also have word representation layers. Word representation initialization is proven helpful in these tasks. Mikolov et al. (Mikolov et al. 2013a) design a test set for evaluating syntactic and semantic regularities preserved in the word representations. They also propose two neural network models for word representation learning: CBOW and Skip-gram. Specifically, a sum pooling layer is employed in CBOW which significantly speeds up the training process. CBOW can be trained over billions of words in one day. The training speed is much faster than the neural network models reported in (Bengio et al. 2003; Collobert and Weston 2008; Collobert et al. 2011), which use sequential projection layers. However, CBOW suffers from the word proximity modeling issue. Skip-gram outperforms previous learning models on representation learning. Mikolov et al. (Mikolov et al. 2013b) further improve the performance and training speed of Skip-gram by employing negative sampling.

From a linguistic perspective, researchers are exploring ways to hand word sense ambiguity in training word representations. Reisinger et al. (Reisinger and Mooney 2010) propose creating multiple "sense-specific" representation vectors for one word. When measuring word similarity without context, they just pick the smallest distance among all word sense vector pairs. They incorporate a clustering algorithm when measuring word similarity with con-

Table 1: Statistics of the POS ambiguity in Wikipedia documents

POS Tag	Word Occurrences Threshold	Non-dominant POS Occurrences	Total Word Occurrences	Ratio	Coverage in All Ambiguous Words
All POS tags	>5	196,795,312	1,632,407,847	12.06%	N/A
	>1,000	188,782,936	1,563,857,888	12.07%	N/A
	>10,000	169,595,926	1,450,251,521	11.69%	N/A
Noun, Verb, Adjective, Adverb	>5	184,256,253	1,034,196,882	17.82%	93.63%
	>1,000	177,753,314	968,408,711	18.36%	94.16%
	>10,000	159,829,386	856,609,611	18.66%	94.24%

text. Huang et al. (Huang et al. 2012) adopt the idea of "sense-specific" representation in their work where the word representation is trained with neural networks.

Proximity-Ambiguity Sensitive Models

We propose the PAS models for producing high quality distributed representations of words by considering both proximity and ambiguity. In both models, we handle POS ambiguity by allowing multiple representation vectors for one word. In PAS CBOW, word proximity is modeled by adding proximity weights to the pooling layer. The proximity weights are learned together with the word representations during training. In PAS Skip-gram, we model the word proximity by using the proximity weights learned with PAS CBOW. We present the two PAS models in this section.

Ambiguity Modeling

In CBOW and Skip-gram (Mikolov et al. 2013a), a word can only have one single representation vector. However, it's common for a word to have multiple valid POS tags, each of which reflects one lexical category the word may belong to. Taking "account" as an example, the verb "account" and the noun "account" have different semantic meanings. It's very hard to capture regularities of the verb "account" and the noun "account" in one representation vector simultaneously. The regularities of the minority POS tags tend to be ignored while the regularities of the majority POS tags are interfered by the minority ones.

POS ambiguity widely exists in natural language texts. Many machine learning algorithms have been applied to assign POS tags with high accuracy, such as Hidden Markov Models (HMM) (Manning and Schütze 1999), Conditional Random Fields (CRF) (Lafferty, McCallum, and Pereira 2001) etc. We train a CRF POS tagger on the Wall Street Journal data from Penn Treebank III (Marcus, Marcinkiewicz, and Santorini 1993). The accuracy of the POS tagger is about 97%. We process all Wikipedia documents (total 1.6 billion words) with the POS tagger. Table 1 shows the statistics of the POS ambiguity in the Wikipedia documents. A POS tag is considered the dominant POS tag of a word if it is assigned to over 90% occurrences of the word. Among normal words (occurrences > 5), the non-dominant POS tag occurrences cover over 12% of the total occurrences. When we look at high frequency words (occurrences $>$ 10,000), the non-dominant POS occurrences still cover over 11% of the total occurrences.

In PAS CBOW and PAS Skip-gram, we train multiple representation vectors for one word. After the training corpus is processed by our POS tagger, each word within it is associated with its POS tag. We alter the words in the training corpus by concatenating a word with its POS tag. For example, the word "account" in the sentence "I have an empty bank account." is changed to "account#NN", where "NN" means noun (full POS tag set can be found in (Marcus, Marcinkiewicz, and Santorini 1993)). In this way, a word can have multiple representation vectors. For instance, the word "account" may have two vectors: $vector(\text{“}account\#NN\text{”})$ and $vector(\text{“}account\#VB\text{”})$.

Besides training one representation vector for each POS tag of a word (we call *Fine-Grained POS*), we also try merging POS tags into groups because the occurrences of a word may hold the same meaning even when their POS tags are different. For example, the word "association" in "Association for Computing Machinery" is tagged as "NNP" (proper noun), but it shares the same meaning with the noun "association". We propose two grouping strategies. The first is called *Coarse-Grained POS* in which there are only 5 groups {N, V, J, R, OTHER}. "N" includes nouns and their variations {NN, NNS, NNP, NNPS}; "V" includes verbs and their variations {VB, VBD, VBG, VBN, VBP, VBZ}; "J" includes adjectives and their variations {JJ, JJR, JJS}; "R" includes adverbs and their variations {RB, RBR, RBS}; "OTHER" includes the rest of the POS tags. *Coarse-Grained POS* is proposed based on the observation that most POS ambiguities (over 93% as shown in Table 1) are among nouns, verbs, adjectives, adverbs and their variations (i.e. plural form, past tense etc.). The second is *Medium-Grained POS* in which there are 14 groups: {{NN, NNP} ,{NNS, NNPS} ,{VB, VBP} ,{VBD} ,{VBN} ,{VBG} ,{VBZ} ,{JJ} ,{JJR} ,{JJS} ,{RB} ,{RBR} ,{RBS} ,OTHER}.

PAS CBOW

In CBOW, the neural network input is the words inside a context window around the output word. The representation vectors of the context words are summated at the sum pooling layer. The output word is represented by a Huffman binary tree in the classification section. The objective function is a hierarchical softmax. Stochastic Gradient Descent (SGD) is used to train CBOW while the gradient is calculated with the *back-propagation* algorithm.

We propose PAS CBOW by adding proximity weights to the sum pooling layer of CBOW as shown in Figure 1. In the figure, $W\#_t$ represents the altered word (word together with its POS tag group, "word" for short) at position t; V_t represents the representation vector of $W\#_t$; the edges

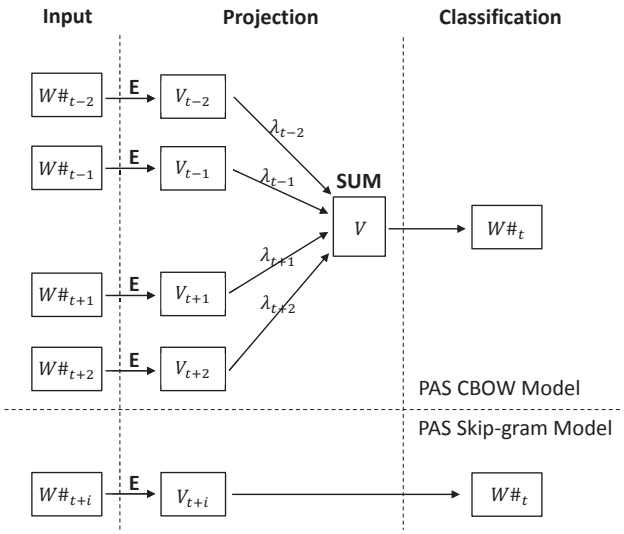

Figure 1: PAS CBOW & PAS Skip-gram

with label "E" represent the representation mapping layers; λ_{t+1} represents the proximity weight. The representation vector of each context word is multiplied by the proximity weight that corresponds to the relative position of the context word. When feeding forward, the projection vector in the PAS CBOW model is

$$V = \sum_{|i|<\tau/2, i\neq 0} \lambda_i V_{t+i} \tag{1}$$

where τ is the window size; λ_i is the proximity weight at the relative position i.

The proximity weights are updated together with the word representations during training. Since the proximity weight multiplication can be regarded as feeding forward a special neural network layer, we can still use the *back-propagation* algorithm to calculate the gradients of the proximity weights. The error signal δ at the projection layer is propagated to each representation layer as

$$\delta_{t+i} = \lambda_i \delta \tag{2}$$

The gradient of the proximity weight λ_i is then

$$\frac{\partial L}{\partial \lambda_i} = V_{t+i} \cdot \delta \tag{3}$$

where L is the objective function; $\cdot$ is a dot product. When training the network with SGD, λ_i is updated together with the word representations. Specifically, the initial values of the proximity weights are "1"s instead of random values.

PAS Skip-gram

Skip-gram is less sensitive to word proximity. In Skip-gram, the neural network input is one single word inside a context window around the output word. When enumerating the word pairs (input and output) in a training corpus, Skip-gram actually captures the total co-occurrences of two words. The local word proximities are averaged over whole training corpus, making Skip-gram less sensitive to the word proximity.

We cannot add proximity weights into Skip-gram as we do in PAS CBOW. The supervision signal for the proximity weights during training PAS CBOW comes from the differences among context words. However, there is only one context word at the input layer of Skip-gram which makes it impossible to learn the proximity weights as in PAS CBOW.

We propose PAS Skip-gram which directly uses the proximity weights learned in PAS CBOW. The proximity weights cannot be multiplied to the word representation vector as we do in PAS CBOW because that would bring scale fluctuation to the projection vector. Instead, we replace the prior of *dynamic window size* with the prior derived from the proximity weights. When applying *dynamic window size* to PAS Skip-gram, only the word inside the selected context window is fed to the input layer. The prior distribution of the window size decides how the word pair (input and output) co-occurrences are averaged. An appropriate prior distribution can improve word representation learning. We scale the proximity weights learned with PAS CBOW to make the summation equal to 1. The normalized weights can be regarded as a pseudo probability distribution. We use the pseudo probability distribution as a prior for *dynamic window size* in PAS Skip-gram.

Experiments

We test the effectiveness of the proximity modeling by comparing the PAS models with CBOW/Skip-gram without considering word ambiguity. We present the experimental results of the different POS tag grouping strategies when considering word ambiguity. We then present the results of the PAS models considering both proximity and ambiguity.

Data Sets

We conduct experiments on two text corpora created from Wikipedia documents. The big one contains 1.6 billion words from all Wikipedia documents while the small one contains 42 million words from a random sample of Wikipedia documents. We use the test set proposed in (Mikolov et al. 2013a) to measure the quality of the learned representation vectors. There are a total of 8,869 semantic and 10,675 syntactic questions in the test set. The semantic questions are categorized into five types while the syntactic questions are categorized into nine types (as shown in Table 2). An example question is "what to 'woman' is like 'king' to 'man'?". The answer to the question is "queen". With the distributed word representation vectors, the question is answered by finding the closest word vector (Euclidean distance) to $vector(\text{“}king\text{”}) - vector(\text{“}man\text{”}) + vector(\text{“}woman\text{”})$. The accuracy, i.e. the ratio of correctly answered questions, is used to measure the quality of the learned distributed word representations.

Proximity Modeling

We try the proximity modeling method in PAS CBOW on the small corpus. Since we only want to compare the proximity modeling capability, word ambiguity is not considered. Three different vector dimensions (50, 300 and 600)

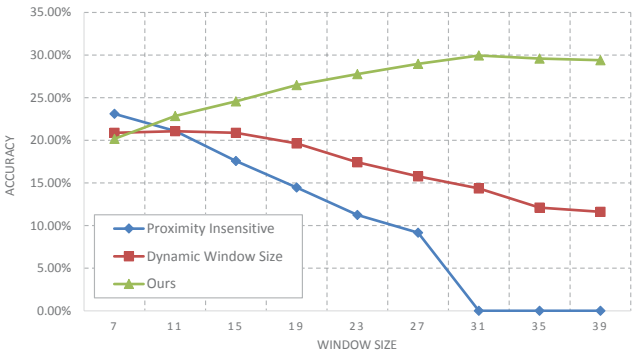

Figure 2: Comparison among different proximity modeling methods in CBOW

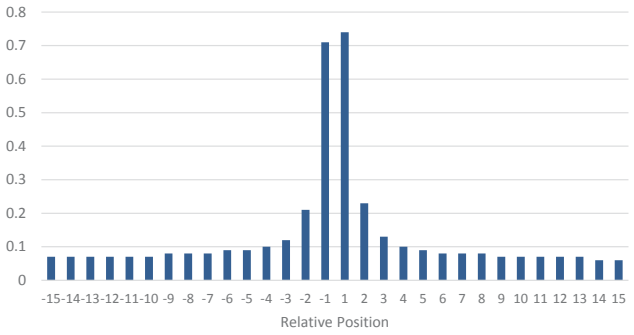

Figure 3: The proximity weights learned from the small corpus

are employed in the experiments. Increasing vector dimension results in better accuracy as reported in (Mikolov et al. 2013a). In the following, we only present the results where vector dimension equals to 600.

Figure 2 shows the comparison among different proximity modeling methods in CBOW. For each method, we train CBOW with different window size settings and evaluate it on the test set. The trend "Proximity Insensitive" corresponds to the proximity insensitive CBOW where the context window size is fixed during training. The accuracy of the proximity insensitive CBOW drops rapidly with the increase of window size since word proximity is totally ignored. When the window size becomes large, the bag-of-words become noisier as the order information is removed (proximity is not modeled). Eventually, CBOW fails to learn meaningful word representations (accuracy goes to zero) when the window size is 31.

By employing *dynamic window size*, the accuracy level drops slower than the proximity insensitive CBOW (see the trend "Dynamic Window Size"). Here the window size means the maximum size that can be drawn from prior distribution. The accuracy drop slows down because *dynamic window size* tends to use a small context window. Also as discussed in the previous section, the *dynamic window size* brings fluctuation to the word representation vector since the scales of the projection vector items are unstable.

We model word proximity by automatically learning the proximity weights. Figure 3 shows the learned proximity weights (the window size is 31). The proximity weight drops rapidly as the context word distance increases, which is very different from the linear prior employed in *dynamic window size*. Since the proximity weights are used to scale the contribution of context words, we can use a fixed context window size. The trend "Ours" in Figure 2 shows the accuracy of CBOW when employing our proximity modeling method. Unlike *dynamic window size*, the accuracy increases as the window size goes up before the size reaches 31. There are two reasons for the accuracy increase. First, adding proximity weights actually helps to recover the order information of the input context words. Second, the window size is fixed during training so that the word representation vectors do not suffer from the scale variation of the projection vector. Moreover, our proximity modeling method can also be used in other neural network applications, which employ pooling layers like sum-pooling, max-pooling etc.

We further test our proximity modeling method on the big corpus. The results are shown in Table 2. The accuracy of the semantic questions increase by 25.9%, the accuracy of the syntactic questions increase 3.1%, while the total accuracy goes up 13.5%. We also test our proximity modeling method in Skip-gram. Although Skip-gram is less sensitive to word proximity, we still achieve a 1.8% overall accuracy increase.

Although additional calculations are introduced by adding proximity weights, the training time does not see a significant increase. For CBOW, the training process is 42% slower due to the additional calculation brought by the proximity weights. However, the training process is 49% faster for Skip-gram since fewer words are involved in the calculation by leveraging the proximity weights learned from CBOW.

Comparison of POS Grouping

We compare the performance of the different POS grouping strategies including *Coarse-Grained POS*, *Medium-Grained POS* and *Fine-Grained POS*. *Fine-Grained POS* treats each POS tag as a group. *Coarse-Grained POS* has 5 groups while *Medium-Grained POS* has 14 groups. The detailed grouping strategy in presented in the previous section. In the comparison experiments, we want to compare the performance of the POS grouping strategies only, so we don't employ our proximity modeling method. *Dynamic window size* is used instead.

Table 3: Comparison of the POS grouping strategies

Strategy	CBOW	Skip-gram
Ambiguity Insensitive	44.61%	58.89%
Fine-Grained POS	43.98%	58.57%
Coarse-Grained POS	45.78%	59.88%
Medium-Grained POS	**45.96%**	**60.21%**

We conduct the experiments on the big corpus, which is processed with our POS tagger. The words inside the corpus are altered based on the POS grouping strategy. We obtain get 3 new corpora from the original big corpus each of which corresponds to a POS grouping strategy. For the test set, we manually decide which representation vector to use for each

Table 2: Proximity modeling on the big corpus

Domains	Without Proximity Modeling	With Proximity Modeling
Semantic	37.22% (3301/8869)	**63.13%** (5599/8869)
capital-common-countries	48.42% (245/506)	82.61% (418/506)
capital-world	52.59% (2379/4524)	75.99% (3438/4524)
currency	7.85% (68/866)	6.12% (53/866)
city-in-state	10.70% (264/2467)	51.24% (1264/2467)
family	68.18% (345/506)	84.19% (426/506)
Syntactic	54.95% (5866/10675)	**58.08%** (6200/10675)
adjective-to-adverb	7.56% (75/992)	5.44% (54/992)
opposite	25.49% (207/812)	31.90% (259/812)
comparative	75.38% (1004/1332)	79.20% (1055/1332)
superlative	37.79% (424/1122)	43.05% (483/1122)
present-participle	46.50% (491/1056)	51.23% (541/1056)
nationality-adjective	86.37% (1381/1599)	87.62% (1401/1599)
past-tense	52.95% (826/1560)	57.63% (899/1560)
plural	68.62% (914/1332)	66.59% (887/1332)
plural-verbs	62.53% (544/870)	71.38% (621/870)
Total	46.90% (9167/19544)	**60.37%** (11799/19544)

question word and each answer word. This is easy to do so since the words in the test set have obvious POS dispositions. Accordingly, we get 3 new test sets from the original test set. We train CBOW and Skip-gram on each training corpus and evaluate it on the corresponding test set. Table 3 shows the results.

The accuracies of both models drop a little bit when employing *Fine-Grained POS*. As discussed in the previous section, the occurrences of a word may hold the same meaning even when their POS tags are different. Learning one representation vector for each POS tag actually divides the word occurrences, where the word holds the same meaning, into small partitions. The representation vector is insufficiently trained on the corresponding partition. Therefore, POS grouping is preferred.

Coarse-Grained POS keeps only five groups because most POS ambiguities are among nouns, verbs, adjectives, adverbs and their variations. It handles the ambiguous POS tags across groups, such as noun/verb ambiguity. By using *Coarse-Grained POS*, both models receive accuracy gain.

Medium-Grained POS is sensitive to subtle POS ambiguities, such as paste tense verb/past participle verb ambiguity. Such POS ambiguities are ignored in *Coarse-Grained POS*. By using *Medium-Grained POS*, more accuracy gain is achieved.

PAS Models

We finish by conducting experiments on PAS CBOW and PAS Skip-gram by considering both proximity and ambiguity. The window size is set to 31 since the accuracy drops a little bit after the size exceeds 31 according to Figure 2. We employ *Medium-Grained POS* in both models according to the experiments in the previous section. The training of each model lasts 3 epoches. The results are shown in Table 4.

It is clear that both models outperform the original models CBOW and Skip-gram. Specifically, PAS CBOW gets an accuracy gain of 16.92% over CBOW. PAS CBOW even outperforms PAS Skip-gram which proves the effectiveness of the proximity modeling on the pooling layer. In the modified Skip-gram reported in (Mikolov et al. 2013b), the hierarchical softmax cost function is replaced by Noise Contrastive Estimation (NCE) which differentiates the output word from a set of randomly sampled words. Our PAS models can also adopt the NCE cost function since both proximity modeling and ambiguity modeling are separated from the cost function selection.

Table 4: PAS performance

Model	Semantic	Syntactic	Total
CBOW	37.22%	54.95%	46.90%
Skip-gram	**70.89%**	50.87%	59.95%
PAS CBOW	62.87%	**64.61%**	**63.82%**
PAS Skip-gram	69.93%	58.44%	63.65%

Conclusion

In this paper, we carry out PAS CBOW and PAS Skip-gram to learn word representations by considering both word proximity and ambiguity. We handle the word ambiguity problem by training multiple representation vectors for one word. We focus on POS ambiguity and further propose different grouping strategies. In PAS CBOW, the word proximity is modeled by adding proximity weights to the sum pooling layer. The proximity weights are automatically learned together with the word representation during training. The proximity modeling can also be applied to other neural network applications, which employ pooling layers. In PAS Skip-gram, we leverage the proximity weights learned from PAS CBOW to handle the proximity issue. With the proposed PAS models, we achieve a maximum accuracy increase of 16.9% over state-of-the-art models in word representation learning.

References

Bengio, Y.; Ducharme, R.; Vincent, P.; and Jauvin, C. 2003. A neural probabilistic language model. *Journal of Machine Learning Research* 3:1137–1155.

Collobert, R., and Weston, J. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, 160–167. ACM.

Collobert, R.; Weston, J.; Bottou, L.; Karlen, M.; Kavukcuoglu, K.; and Kuksa, P. 2011. Natural language processing (almost) from scratch. *The Journal of Machine Learning Research* 12:2493–2537.

Elman, J. L. 1991. Distributed representations, simple recurrent networks, and grammatical structure. *Machine learning* 7(2-3):195–225.

Hinton, G. E. 1986. Learning distributed representations of concepts. In *Proceedings of the eighth annual conference of the cognitive science society*, 1–12. Amherst, MA.

Huang, E. H.; Socher, R.; Manning, C. D.; and Ng, A. Y. 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*, 873–882. Association for Computational Linguistics.

Lafferty, J.; McCallum, A.; and Pereira, F. C. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. *ICML*.

Manning, C. D., and Schütze, H. 1999. *Foundations of statistical natural language processing*, volume 999. MIT Press.

Marcus, M. P.; Marcinkiewicz, M. A.; and Santorini, B. 1993. Building a large annotated corpus of english: The penn treebank. *Computational linguistics* 19(2):313–330.

Mikolov, T.; Chen, K.; Corrado, G.; and Dean, J. 2013a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G.; and Dean, J. 2013b. Distributed representations of words and phrases and their compositionality. *arXiv preprint arXiv:1310.4546*.

Mnih, A., and Hinton, G. 2007. Three new graphical models for statistical language modelling. In *Proceedings of the 24th international conference on Machine learning*, 641–648. ACM.

Mnih, A., and Hinton, G. E. 2008. A scalable hierarchical distributed language model. In *Advances in neural information processing systems*, 1081–1088.

Mnih, A., and Teh, Y. W. 2012. A fast and simple algorithm for training neural probabilistic language models. *arXiv preprint arXiv:1206.6426*.

Morin, F., and Bengio, Y. 2005. Hierarchical probabilistic neural network language model. In *Proceedings of the international workshop on artificial intelligence and statistics*, 246–252.

Reisinger, J., and Mooney, R. J. 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*, 109–117. Association for Computational Linguistics.

Rumelhart, D. E.; Hintont, G. E.; and Williams, R. J. 1986. Learning representations by back-propagating errors. *Nature* 323(6088):533–536.

Schwenk, H., and Gauvain, J.-L. 2004. Neural network language models for conversational speech recognition. In *INTERSPEECH*.

Schwenk, H.; Dchelotte, D.; and Gauvain, J.-L. 2006. Continuous space language models for statistical machine translation. In *Proceedings of the COLING/ACL on Main conference poster sessions*, 723–730. Association for Computational Linguistics.

Schwenk, H. 2007. Continuous space language models. *Computer Speech & Language* 21(3):492–518.

Weston, J.; Bengio, S.; and Usunier, N. 2011. Wsabie: Scaling up to large vocabulary image annotation. In *Proceedings of the Twenty-Second international joint conference on Artificial Intelligence-Volume Volume Three*, 2764–2770. AAAI Press.