Locating and Filling Missing Words in Sentences Based on Language Models

Tianlong Song Zhe Wang
Dept. of Electrical & Computer Engineering, Michigan State University
East Lansing, MI 48824, USA
{songtia6,wangzh34}@msu.edu

Abstract

This report considers locating and filling missing words in sentences based on language modeling. In locating the missing words, two approaches are proposed, which are based on N-gram models and word distance statistics (WDS), respectively. In filling the missing words, we first reduce the candidate word space by restricting our attention to only the words that have previously showed neighborhood with either of the two immediately surrounding words at the missing word location, and then identify the most probable missing word by measuring the statistical connection between the candidate words and the surrounding words. It is shown that for both missing word location and missing word filling, the proposed approaches achieve much higher accuracies than chance.

1 Introduction

1.1 Problem Statement

There has been many occasions that we have incomplete sentences that are needed to completed. One example is that in speech recognition noisy environment can lead to unrecognizable words, but we still hope to recover and understand the complete sentence (e.g., by inference); another example is sentence completion questions that appear in language tests (e.g., SAT, TOEFL, GRE, etc.). Generally, the problem we are aiming to solve is locating and filling any missing words in incomplete sentences. However, this problem seems too ambitious so far, and we direct ourselves to a simplified version of this problem.

To simplify the problem, we assume that there is only one missing word in a sentence, and the missing word is neither the first word nor the last word of the sentence. This problem originally comes from *https://www.kaggle.com/c/billion-word-imputation*. In this project, we propose to locate and then fill the missing word for given incomplete sentences using computational approaches based on language modeling. More specifically, we first train our model by complete sentences, and then apply the trained model to locate and fill missing word in new incomplete sentences.

1.2 Related Work

Chelba et al. (2013) from Google proposed a new benchmark corpus for measuring statistical language modeling. They show performance of several well-known types of language models. The baseline unpruned Kneser-Ney 5-gram model achieves perplexity 67.6, and a combination of techniques leads to a 35% further reduction in perplexity.

Mccarthy et al. (2007) studied the English lexical substitution task. The task involved both finding the synonyms and disambiguating the context. Eight teams participated this task, among which Yuret and Hassan et al. outperformed the others. Yuret (2007) adopted the simple 5-gram model to evaluate the likelihood of various substitutes for a word in a given context. These likelihoods were then used to determine the best sense for the word in novel contexts. Hassan et al. (2007) combined several techniques and knowledge sources to provide the most likely set of substitutes for a word in a given context, including WordNet, Microsoft Encarta encyclopedia

and Roget thesaurus. Methods involved in ranking all candidate word include Lexical Baseline (LB), Machine Translation (MT), Most Common Sense (MCS) and Latent Semantic Analysis (LSA).

Turney (2001) studied the synonym questions in English exams. He presented a simple unsupervised learning algorithm for recognizing synonyms, based on statistical data acquired by querying a Web search engine. Jarmasz and Szpakowicz (2003) correlated several different systems and got a state-of-art performance 82%. Zweig et al. (2012) studied the sentence completion problems in SAT exams. They investigated several different approaches, including N-gram and LSA, and found that both models did significantly better than chance, and could be even more effective when they were combined.

2 Proposed Approaches

The data sets we use come from *https://www.kaggle.com/c/billion-word-imputation/data*. The training set contains a large collection of English language sentences. In the testing set, each entry is an incomplete sentence with exactly one word removed. The location of the removed word was chosen uniformly randomly and is never the first or last word of the sentence. The problem is solved in two steps: locating the missing word and then filling the missing word.

2.1 Locating the Missing Word

2.1.1 N-gram Model

For a given training data set, define $C(w_1, w_2)$ as the number of occurrences of the bigram pattern (w_1, w_2), and $C(w_1, w, w_2)$ the number of occurrences of the trigram pattern (w_1, w, w_2). Then, the number of occurrences of the pattern, where there is one and only one word between w_1 and w_2, can be calculated by

$$D(w_1, w_2) = \sum_{w \in V} C(w_1, w, w_2), \qquad (1)$$

where V is the vocabulary.

Consider a particular location, l, of an incomplete sentence of length L, and let w_l be the lth word in the sentence. $D(w_{l-1}, w_l)$ would be the number of positive votes from the training data set for missing word at this location, while $C(w_{l-1}, w_l)$ would be correspondingly the number of negative votes. We define the score indicating there is a missing word at location l as

$$\begin{aligned} S_l =& \frac{D(w_{l-1}, w_l)^{1+\gamma}}{C(w_{l-1}, w_l) + D(w_{l-1}, w_l)} \\ & - \frac{C(w_{l-1}, w_l)^{1+\gamma}}{C(w_{l-1}, w_l) + D(w_{l-1}, w_l)}, \end{aligned} \qquad (2)$$

where γ is a small positive constant. Hence, the missing word location can be identified by

$$\hat{l} = \arg\max_{1 \le l \le L-1} S_l. \qquad (3)$$

Note that in (2), if we set $\gamma = 0$, the upper part would be exactly the percentage of positive votes for missing word at that location, and the lower part is the percentage of negative votes. It seems a fairly reasonable score, then *why do we still need a positive* γ? The underlying reason is that intuitively the more number of votes for a particular decision, the more confident we are on that decision. This trend is reflected in a positive γ, which can be viewed as *sparse vote penalty* and is useful in breaking ties in the missing word location voting. That is, if we have exactly the same ratio of positive votes relative to negative votes for two candidate locations, e.g., 80 positive votes v.s. 20 negative votes for location A, and 8 positive votes v.s. 2 negative votes for location B, we would believe that location A is more likely to be the missing word location compared with location B.

2.1.2 Word Distance Statistics (WDS)

In view of the fact that the statistics of the two words immediately adjacent to a given location contribute a lot in deciding whether the location has a word missing, we tentatively guess that all the words within a window centered at that location would more or less contribute some information as well. To incorporate statistics on more words that are related, we generalize the idea in Section 2.1.1 by investigating a word window instead of only the two immediately adjacent words. To this end, we introduce the concept of word distance statistics (WDS).

More specifically, we use $\widetilde{C}(w_1, w_2, m)$ to denote the number of occurrences of the pattern, where there is exactly m words between w_1 and w_2, i.e., the word distance of w_1 and w_2 is m. For a given

location l in an incomplete sentence and a word window size W, we are interested in the word distance statistics of each word pair, in which one word w_i is on the left of the location l, and the other word w_j is on the right, as illustrated in Figure 1.

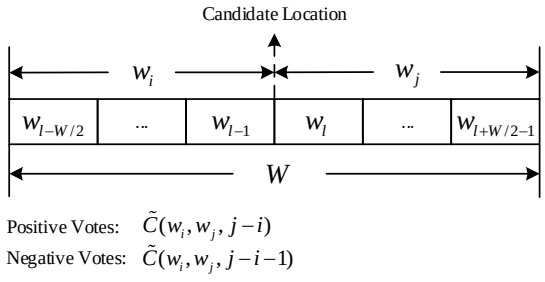

Figure 1: Word distance illustration.

Formally, for any $l - W/2 \leq i \leq l - 1$ and $l \leq j \leq l + W/2 - 1$, $\widetilde{C}(w_i, w_j, j - i)$ would be the number of positive votes for missing word at this location, while $\widetilde{C}(w_i, w_j, j - i - 1)$ is the number of negative votes. Applying the idea in (2), for each word pair (w_i, w_j), we extract its feature as the score indicating there is a missing word at location l, i.e.,

$$S_l(i,j) = \frac{\widetilde{C}(w_i, w_j, j-i)^{1+\gamma}}{\widetilde{C}(w_i, w_j, j-i) + \widetilde{C}(w_i, w_j, j-i-1)} - \frac{\widetilde{C}(w_i, w_j, j-i-1)^{1+\gamma}}{\widetilde{C}(w_i, w_j, j-i) + \widetilde{C}(w_i, w_j, j-i-1)}. \quad (4)$$

As a special example, let $i = l - 1$ and $j = l$, (4) would be reduced to (2).

To find the missing word location, we need to assign different weights to the extracted features, $S_l(i, j)$. Then, the missing word location can be determined by

$$\hat{l} = \arg\max_{1 \leq l \leq L-1} \sum_{l - \frac{W}{2} \leq i \leq l-1} \sum_{l \leq j \leq l + \frac{W}{2} - 1} v(i,j) S_l(i,j), \quad (5)$$

where the weight, $v(i, j)$, should be monotonically decreasing with respect to $|j - i|$.

2.1.3 Weight Learning by Logistic Regression

To learn the weights in (5), we apply the logistic regression model. More specifically, consider a location within a given incomplete sentence, let "1" denote that there is one word missing at this location, while "-1" denotes that there is no word missing here. We train a binary logistic classifier that can provide a binary result indicating a missing word or not as well as the corresponding confidence. Then, the missing word location can be determined as the one with highest confidence.

To generate training examples, for each complete sentence in the training set, we randomly choose a location l, remove the word at this location, calculate the feature vector $\mathbf{f}$ according to (4), and then formulate the positive training example as $(\mathbf{f}, 1)$, where "1" indicates that there is a missing word at this location. A negative training example, $(\mathbf{f}, -1)$, can be generated similarly but without any words removed.

Although a lot of effort is made here, unfortunately no obvious performance improvement has been achieved by the weights learned by logistic regression compared to an empirical weight setting. The possible reasons include: (1) the missing word location accuracy is dominated by the two words immediately adjacent to the location, so better weight choices probably make little difference in terms of performance enhancement; (2) the weight trend (monotonically decreasing with respect to the word distance) is straightforward and clear, so a good empirical setting would probably be already good enough.

2.2 Filling the Missing Word

Filling the missing word would be an even more challenging task than locating the missing word. This is because for missing word location, the candidate locations are limited; while for missing word filling, the candidate word space may be as large as the entire vocabulary. In light of this observation, we try to reduce the candidate word space, which is done in two steps.

First, we explore the training sentences and identify a set of high-frequency words, V_h, which can cover most of word occurrences. Second, we investigate different word groups within the high-frequency word set V_h in terms of covering the ground-truth words to be filled. More specifically, for a true missing word location l in an incomplete sentence, in which the word sequence is "..., $w_{l-2}, w_{l-1}, w_l, w_{l+1}$, ...", the three word groups we investigate are: (1) Group A contains any word

$w \in V_h$, such that the trigram (w_{l-1}, w, w_l) occurs at least once in the training data; (2) Group B contains any word $w \in V_h$, such that either the bigram (w_{l-1}, w) or (w, w_l) occurs at least once in the training data; (3) Group C contains group B plus any other word $w \in V_h$, such that the trigram $(w_{l-2}, *, w)$ or $w, *, w_{l+1})$ occurs at least once in the training data, where "*" denotes an arbitrary word. Apparently, we have $A \subseteq B \subseteq C$. Table 1 shows the percentages of different word groups in terms of covering the ground-truth words to be filled. It is observed that group B provides a pretty good coverage, and extending to group C results in little gain. As a result, in the following, we consider group B as our candidate word space instead of the entire vocabulary.

Word Group	A	B	C
Coverage	35.20%	86.75%	88.35%

Table 1: Percentages of different word groups in terms of covering the ground-truth words to be filled.

To find the most probable word in the given missing word location, we take into account five conditional probabilities, as shown in Table 2, to explore the statistical connection between the candidate words and the surrounding words at the missing word location. Ultimately, the most probable missing word can be determined by

$$\hat{w} = \arg\max_{w \in B} \sum_{1 \le i \le 5} v_i P_i, \qquad (6)$$

where the weight v_i is used to reflect the importance of each conditional probability in contributing to the final score.

	Weight	Definition
P_1	1.0	$P(w\|w_{l-1} * w_l) = \frac{Cnt(w_{l-1} w w_l)}{Cnt(w_{l-1} * w_l)}$
P_2	0.5	$P(w\|w_{l-2} w_{l-1} *) = \frac{Cnt(w_{l-2} w_{l-1} w)}{Cnt(w_{l-2} w_{l-1})}$
P_3	0.5	$P(w\| * w_l w_{l+1}) = \frac{Cnt(w w_l w_{l+1})}{Cnt(w_l w_{l+1})}$
P_4	0.25	$P(w\|w_{l-1} *) = \frac{Cnt(w_{l-1} w)}{Cnt(w_{l-1})}$
P_5	0.25	$P(w\| * w_l) = \frac{Cnt(w w_l)}{Cnt(w_l)}$

Table 2: Conditional probabilities considered in missing word filling. Note that in the table "*" denotes an arbitrary word.

2.3 Failed Trial: POS Tagging

We had been considering that POS tag information might be helpful in either locating or filling the missing word. As a result, we tried to tag each sentence and then reused our proposed approaches by replacing words with tags. However, the results of using POS tags are much worse than using words instead. The underlying reasons, according to our analysis, include: (1) there are errors in POS tagging for incomplete sentences, especially for the word immediately after the removed word; (2) the POS tags for even incomplete sentences are misleading, since the tagger still produces the most probable tag path, which always contains strong connection between neighboring tags, even at missing word locations. This leads to difficulties in the "anomaly" detection at missing word locations by our approaches.

3 Experimental Evaluation

In this section, we evaluate the performance of our proposed approaches by cross validation. The training data contains $30,301,028$ complete sentences, of which the average sentence length is approximately 25. In the vocabulary with a size of $2,425,337$, $14,216$ words that have occurred in at least 0.1% of total sentences are labeled as high-frequency words, and the remaining $58,417,315$ words are labeled as 'UNKA'. To perform the cross validation, in our experiments, the training data is splitted into two part, TRAIN and DEV. The TRAIN set is used to train our models, and the DEV set is applied to test our models.

3.1 Experimental Results

3.1.1 Missing Word Location Using Different Approaches

Table 3 shows the estimation accuracy of the missing word locations for the two proposed approaches, N-gram and WDS. For comparison, we list the corresponding probabilities by chance as well. Each entry shows the probabilities that the correct location is included in the ranked candidate location list returned by each approach, where the list size varies from 1 to 10. The sparse vote penalty coefficinet, γ, is set to 0.01. In the WDS approach, we consider a word window size $W = 4$. Hence, four pairs of words are taken into account, and the em-

pirical weight settings for them are shown in Table 4.

	Chance	N-gram	WDS
Top 1	4%	51.47%	52.06%
Top 2	8%	63.70%	64.50%
Top 3	12%	71.00%	71.76%
Top 5	20%	80.26%	80.91%
Top 10	40%	91.54%	91.93%

Table 3: Accuracy of Location

$\|j-i\|$	1	2	3
weight	1	0.1	0.01

Table 4: Weights configuration for $S_l(i,j)$

It is demonstrated that both the two proposed approaches, N-gram and WDS, perform significantly better than chance, and a small performance gain is observed for WDS relative to N-gram. Taking the "Top 1" row as an example, which menas that only the candidate location with the highest score is returned, the N-gram approach achieves an accuracy of 51.47%, and the WDS approach achieves an accuracy of 52.06%. When we extend the size of the ranked location to 10, the true missing word locations can be covered with a probability of over 91%.

3.1.2 Accuracy v.s. Number of training samples

Table 5 shows the changes of the accuracy from the N-gram approach with the size of the training data. The parameter γ is set to be 0.01 in all experiments. As expected, larger size of training data leads to higher classification accuracy. When only 0.1 million sentences are used, the N-gram approach can only achieve an accuracy of 36.11%. That number is increased to 51.47% after 30 million sentences are used.

On the other hand, larger size of training data also requires higher occupation of physical memory and longer training time. In our experiments, it took only 13 minutes to train 5 million sentences but 2 hours to train 30 million sentences.

3.1.3 Missing Word Filling

Table 6 shows the accuracies of filling the missing word given the location. Each row of the second column shows the probability that the correct word is included in the ranked candidate words list returned by the proposed approach. The setting of this step is illustrated in section 2.2. It is observed that, when considering only the candidate word with the highest score, the accuracy is 32.15%. After taking more candidates into consideration, that probability that the correct word is covered in the returned list is increased to be above 50%. For the top 10 candidates, the number is increased to 59.15%. Compared to the pure chance, this probability has been considerably high.

Number (million)	Accuracy
0.1	36.11%
0.5	41.06%
1	42.56%
2	43.95%
5	48.32%
30	51.47%

Table 5: Accuracy v.s. Training Data Size

	Accuracy
Top 1	32.15%
Top 2	41.49%
Top 3	46.23%
Top 5	52.02%
Top 10	59.15%

Table 6: Accuracy of Missing Word Filling

3.2 Discussion

To the best of our knowledge, there are not much existing work to which our approaches can be directly compared. The only work that is similar to ours was done by Zweig et al. (2012), who focused on automatically filling blanks in ETS sentence completion problems. They reported that their final accuracy for words filling was about $52\% \sim 53\%$. However, the problem they dealt with is different from ours. First, in their problems, the locations of the missing words have already been known. Second, for the word filling step, the candidate words are limited to no more than 5 options. That means, even random selection could achieve an accuracy of 20%.

Comparing with this paper, our task is much more

challenging. On one hand, the locations of the missing word in our problem need to be identified; on the other hand, the amount of possible candidate words in each location is huge, even after we take into consideration only the high-frequency words. In view of the observations above, the results we reported are quite promising.

4 Conclusions

In this report, we considered locating and filling missing words in sentences based on language modeling. In locating the missing words, we proposed two approaches, which are based on N-gram models and word distance statistics (WDS), respectively. In filling the missing words, we first reduced the candidate word space by restricting our attention to only the words that have previously showed neighborhood with either of the two immediately surrounding words at the missing word location, and then identified the most probable missing word by measuring the statistical connection between the candidate words and the surrounding words. It was shown that for both missing word location and missing word filling, the proposed approaches achieve much higher accuracies than chance.

References

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, and Phillipp Koehn. 2013. One billion word benchmark for measuring progress in statistical language modeling. *CoRR*, abs/1312.3005.

Samer Hassan, Andras Csomai, Carmen Banea, Ravi Sinha, and Rada Mihalcea. 2007. Unt: Subfinder: Combining knowledge sources for automatic lexical substitution.

Mario Jarmasz and Stan Szpakowicz. 2003. S.: Roget?s thesaurus and semantic similarity. In *In: Proceedings of the RANLP-2003*, pages 212–219.

Diana Mccarthy, Falmer East Sussex, and Roberto Navigli. 2007. Semeval-2007 task 10: English lexical substitution task. In *In Proceedings of the 4th workshop on Semantic Evaluations (SemEval-2007)*, pages 48–53.

Peter Turney. 2001. Mining the web for synonyms: Pmi-ir versus lsa on toefl.

Deniz Yuret. 2007. Ku: Word sense disambiguation by substitution. In *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*, pages 207–214. Association for Computational Linguistics.

Geoffrey Zweig, John C. Platt, Christopher Meek, Christopher J. C. Burges, Ainur Yessenalina, and Qiang Liu. 2012. Computational approaches to sentence completion. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers - Volume 1*, ACL '12, pages 601–610, Stroudsburg, PA, USA. Association for Computational Linguistics.