An Effective Approach for Searching Closest Sentence Translations from The Web

Ju Fan, Guoliang Li, and Lizhu Zhou

Department of Computer Science and Technology
Tsinghua University, Beijing 100084, China
fan-j07@mails.thu.edu.cn; {liguoliang,dcszlz}@thu.edu.cn

Abstract. There are large numbers of well-translated sentence pairs on the Web, which can be used for translating sentences in different languages. It is an interesting problem to search the closest sentence translations from the Web for high-quality translation, which has attracted significant attention recently. However, it is not straightforward to develop an effective approach, as this task heavily depends on the effectiveness of the similarity model which is used to quantify the similarity between two sentences. In this paper, we propose several optimization techniques to address this problem. We devise a phrase-based model to quantify the similarity between two sentences. We judiciously select high-quality phrases from sentences, which can capture the key features of sentences and thus can be used to quantify similarity between sentences. Experimental results show that our approach has performance advantages compared with the state-of-the-art sentence matching methods.

1 Introduction

With the rapid growth of the Web, hundreds of millions of *parallel sentences* (i.e., sentences with their well-translated counterparts) are now available on the Web, forming a rich source for translation reference. The following English sentence with its Chinese translation from a Chinese Web page is an example (the translation is in a pair of parentheses): "The president says he knows the problem too well (总统说他非常了解这个问题)". Recently, utilizing the parallel sentences from the Web to build a sentence-level translation-aid (STA) system for producing high-quality translations has attracted much attention. Some commercial search engines, e.g., Youdao (http://dict.youdao.com) and Iciba (http://dj.iciba.com), have incorporated STA systems into their services.

An overview of a typical STA system is shown in Figure 1. A huge amount of parallel sentences are crawled, extracted from the Web, and stored in a parallel-sentence database. Given a source sentence to be translated by a translator, the Sentence Matching component searches for the most similar (or closest) sentences with their translations from the database. The translator then chooses some of the retrieved sentences, revises their translation counterparts if necessary, and then obtains the appropriate translation for the source sentence by limited revisions. For example, given a source sentence, "My elder sister is one year older than her husband", the Sentence Matching component may find the following two sentences: 1) "My elder sister is one year older than me (我姐姐比我大年龄一岁)", and 2) "My elder sister is older than her husband (我

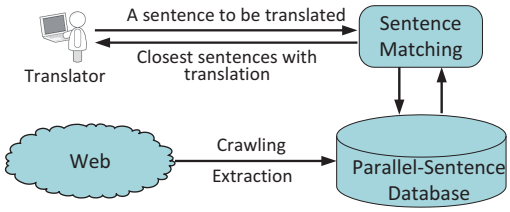


Fig. 1. An overview of a typical STA system.

姐姐比他丈夫年龄大)". Although neither of the retrieved sentences is identical with the source sentence, the translator can reuse these two sentences to obtain the correct translation of the source sentence, "我姐姐比他丈夫年龄大一岁" from either of the Chinese translations of the two sentences with a very minor revision. Some translator surveys have shown that the STA systems are very effective and helpful to both translators and translation companies [5].

In this paper, we study the problem of sentence matching, i.e., searching the closest sentences with their translations for a source sentence, given a parallel-sentence database obtained from the Web. Note that we focus on English-to-Chinese parallel sentences with English as the source sentences to be translated for illustration purpose. Notice that it is not straightforward to develop a sentence matching approach with good performance, as the task heavily depends on the effectiveness of the similarity model that quantifies the similarity between two English sentences. Word-based models, e.g., Translation Models [10, 12], the percentage of shared words [1] and edit distance [13], assume that words in the sentence are mutually independent, and thus they cannot capture the order of words and will lead to low performance. N-gram model [4], one of the Markov models, segments a sentence into a sequence of N-length grams, and makes use of the local dependencies among words. However, it does not consider the syntactic information, which is rather crucial for sentence matching for translation. Cancedda et al. [3] propose the word sequence kernel, which selects all subsequences of sentences and computes the inner product on top of the subsequences to measure the similarity. Although this model has the sound theoretical background, it is very expensive to select all subsequences. To address this problem, we devise a phrase-based similarity model between the source sentence to be translated and the sentences in the database, where a phrase is a subsequence of a sentence. We judiciously select high-quality phrases by considering both the dependency syntax [9] and the frequencies of phrases. The selected phrases can capture the syntactic and frequency features of sentences and thus can be used for effective sentence matching.

The contributions of our paper are summarized as follows: 1) We propose an effective method for searching sentence translations from the Web. 2) We introduce a phrase-based similarity model and propose a method to judiciously select the high-quality phrases. 3) We conducted extensive experiments on two real data sets from the Web, and the experimental results show that our method outperforms existing methods.

The rest of this paper is organized as follows. The related work of sentence matching techniques is briefly reviewed in Section 2. We discuss our similarity model in Section 3 and the phrase-selection algorithms in Section 4. In Section 5 we report our experiments, and finally we conclude in Section 6.

2 Related Work

Sentence matching, a.k.a., sentence retrieval, which retrieves sentences for some requirements, has been widely studied in the research community. Word-based sentence matching methods assume that words in the sentence are mutually independent [12, 10], which is only a simplification in mathematics. Some studies discussed the relationship of words. Metzler and Croft [11] examined the sequential dependencies of words using a Markov assumption. However, the Markov assumption is also too strong. Therefore, some researchers used a syntax parser to analyze the structure of each sentence [6, 2], and then computed the dependencies of words. Some similarity measures combine the above-mentioned ideas. They used sequences of words (i.e., phrases) to replace words by taking into account the order of words. The words in a phrase are closely dependent with each other, and phrases are assumed to be mutually independent. Cancedda et al. [3] proposed the word sequence kernel which extracts all n-length subsequences. Li et al. [7] proposed *VGRAM* (variable-length grams) to select high-quality grams to represent an string (This technique can also been applied to sentences). Hiroshi Ichikawa et al. examined syntactic impacts and manually annotated the syntactic structure of a sentence [6], which is not capable of a large number of sentences.

Our approach introduces the syntactic and frequency information to the similarity measure, and can be seen as an extension of the VGRAM based method [7] and word sequence kernel [3]. On the one hand, we generalize the variable-length grams to the phrases by allowing gaps between words. On the other hand, we incorporate the *syntactic gap constraint*, the *maximum constraint* and the *frequency constraint* to select high-quality phrases to avoid the exponential combination of words in [3].

3 Similarity Models for Sentence Matching

Sentence matching essentially involves a similarity model between a source sentence s to be translated and a sentence u in the parallel-sentence database $\mathcal{U}$. As we use an ordered set of phrases/words to represent a sentence, we use the similarity between two ordered sets to quantify the similarity between the two sentences. Given a source sentence to be translated, we compute the similarity score of every sentence in $\mathcal{U}$ to the source sentence, and suggest the sentences with the highest scores. In this paper, we concentrate on the effectiveness of sentence matching, and take the efficient matching problem as a future work.

Commonly used similarity models between two ordered sets are Jaccard coefficient, cosine distance, and edit distance. Note that we use the sentences in Table 1 as a running example. Suppose we obtain three parallel sentences, u_1, u_2 and u_3 from the Web, and store them in the database. We take s as a source sentence to be translated.

We firstly introduce the three commonly used similarity models. The Jaccard coefficient between s and u is defined as $Sim_j(s,u) = (\mid \mathcal{S}_s \cap \mathcal{S}_u \mid)/(\mid \mathcal{S}_s \cup \mathcal{S}_u \mid)$, where $\mathcal{S}_s$ ($\mathcal{S}_u$) denotes the set of phrases of sentence s (u). Cosine distance is summarized as $Sim_c(s,u) = (\sum_{\bar{f}_i \in \mathcal{S}_s \cap \mathcal{S}_u} w(\bar{f}_i))/(\sqrt{\mid \mathcal{S}_s \mid} \cdot \sqrt{\mid \mathcal{S}_u \mid})$, where $\bar{f}_i$ is an element in the ordered set, $\mathcal{S}_s \cap \mathcal{S}_u$, and $w(\bar{f}_i)$ is the weight of $\bar{f}_i$. The edit distance between two ordered sets is the minimum number of operations, such as insertion, deletion, and substation, required to transform one set into the other.

Table 1. An example (s is the source sentence and $u_1 \sim u_3$ are parallel sentences in the database).

	Sentence	Translation
s	He has eaten an apple	–
u_1	He has an apple	他有一个苹果
u_2	He ate a red apple	他吃了一个红苹果
u_3	He has a pencil	他有一支铅笔

The three models focus on common phrases/words but cannot capture the syntactic information of a sentence. Alternatively, we propose a phrase-based similarity model in Equation (1). The model is based on two factors. One is the percentage of shared phrases, and the other is the sum of their syntactic weights.

$$Sim_p(s,u) = \frac{|\mathcal{S}_s \cap \mathcal{S}_u|}{|\mathcal{S}_u|} \cdot \sum_{\bar{f}_i \in \mathcal{S}_s \cap \mathcal{S}_u} \phi(s,\bar{f}_i)\phi(u,\bar{f}_i)w(\bar{f}_i) \tag{1}$$

where $\frac{|\mathcal{S}_s \cap \mathcal{S}_u|}{|\mathcal{S}_u|}$ is the percentage of the shared phrases between sentences s and u, $\phi(s,\bar{f}_i)$ (or $\phi(u,\bar{f}_i)$) is the *importance* of $\bar{f}_i$ to the sentence s (or u), and $w(\bar{f}_i)$ is the weight of $\bar{f}_i$.

As phrase $\bar{f}_i$ can be discontinuous, its *importance* to s contains two factors: the matched words and the words in *gaps* which are composed of the unmatched words in the phrase. Take a sentence "`He has eaten an apple`" and a phrase "`has apple`" as an example. "`eaten an`" is a gap. We use λ_α to denote the syntactic weight of a matched word α and λ_β to denote the penalty of an unmatched word β in gaps. $\phi(s,\bar{f}_i)$ can be computed as $\phi(s,\bar{f}_i) = \prod_{1 \leqslant j \leqslant m} \lambda_{\alpha_j} \prod_{1 \leqslant k \leqslant n} \lambda_{\beta_k}$, where m is the number of matched words and n is the number of unmatched words in gaps.

Apparently, our model Sim_p is a generalization of cosine distance and Jaccard coefficient, and incorporates the syntactic feature $\phi(s,\bar{f}_i)$ (or $\phi(u,\bar{f}_i)$) into the computation of the similarity. We will experimentally prove that our proposed similarity model achieves the best performance in Section 5.

Now we discuss how to estimate the parameters introduced in the proposed model. Since λ_α is the syntactic weight of a matched word α, we employ the dependency syntax [9] to estimate this weight. In the dependency syntax, a sentence is parsed as a tree structure, where each node is a word in the sentence and an edge between the parent node and the child node represents a *dependency relationship* between the two corresponding words. More specifically, the child word is the modifier of the parent word. In particular, the root of the tree is the predicate of the sentence. For example, consider a sentence "`He has eaten an apple`'. In the parsed tree, the root is the predicate, "`eat`", and the root has three child nodes, i.e., "`he`", "`have`", and "`apple`". In addition, "`an`" is the child node of "`apple`", as the former is the article of the latter. We assume that the ancestors are more important than the decedents in terms of syntax. According to this rule, we introduce a decay factor d ($0 < d \leqslant 1$) to ensure that the weight of a child α_c is smaller than its parent α_p by the decay factor, i.e., $\lambda_{\alpha_c} = d \cdot \lambda_{\alpha_p}$. On the other hand, λ_β is set to a constant for penalizing the unmatched words.

In addition, given a phrase, we use its inverse document frequency (IDF) in $\mathcal{U}$ to estimate its weight, i.e., $w(\bar{f}_i) = \log \frac{|\mathcal{U}|}{|\{u_j | \bar{f}_i \in u_j\}|}$ where $|\mathcal{U}|$ is the size of $\mathcal{U}$, and $\bar{f}_i \in u_j$ means that u_j ($u_j \in \mathcal{U}$) contains the phrase $\bar{f}_i$.

4 High-Quality Phrase Selection

In this section, we discuss an effective method to select high-quality phrases from sentences. We firstly give the formulation of high-quality phrases in Section 4.1, and then develop efficient algorithms to generate such phrases from sentences in Section 4.2.

4.1 High-Quality Phrases

As selecting all subsequences of a sentence as phrases is rather expensive, we propose a heuristic approach to select *the infrequent phrases with syntactic gap constraints* as the high-quality phrases. We restrict the gaps of discontinuous phrases with syntactic rules: if two discontinuous words of a sentence are selected, they must have syntactic dependency relationships (Section 3). Consider the sentence "`he has eaten an apple`" and the phrase "`he eaten an apple`" that has a gap "`has`". The phrase is meaningful because "`he`" and "`eaten`" have a dependency relationship. In contrast, "`has apple`" should be eliminated because "`has`" is independent on "`apple`". We also select phrases according to its frequency. Phrases with high frequencies have small IDF scores, thus they are less important than the infrequent ones according to our similarity model. Therefore we eliminate them in order to improve the efficiency.

Now, we formally define the high-quality phrase. Let a sentence s be a word sequence $s = s_1 s_2 \dots s_{|s|}$, where $| s |$ is the length of s and s_i is the i-th word of s ($1 \leqslant i \leqslant | s |$). Notice that all words are stemmed and stop-words are eliminated (We maintain a subset of commonly used stop-word list where some playing important roles in syntax, i.e., "he" and "I", are excluded). We denote a phrase as $\bar{f}[I] = s_{i_1} s_{i_2}, \dots, s_{i_n}$ where $I = [i_1, i_2, \dots, i_n]$ ($1 \leqslant i_1 < i_2 < \cdots < i_n \leqslant | s |$). The infrequent phrase with syntactic gap constraints (i.e., high-quality phrases) is defined in Definition 1.

Definition 1 (High-Quality Phrases). *$\bar{f}[I]$ is a high-quality phrase if it satisfies:*

1. Gap constraint: If $i_{j+1} \neq i_j + 1$ ($1 \leqslant j < n$), $s_{i_{j+1}}$ and s_{i_j} must have a dependency relationship, and

2. Frequency constraint: $\bar{f}[I]$ is infrequent, that is, its frequency is smaller than a given threshold which is determined empirically, and

3. Maximum constraint: $\bar{f}[I]$ is not a prefix of any other phrases.

Take the sentence "`he ate a red apple`" in Table 1 as an example. We preprocess it into "`he eat red apple`". Suppose that the given frequency threshold is 2. Consider the phrase "`eat apple`". Firstly, it satisfies the gap constraint: "`apple`" is the object of the verb "`eat`". Secondly, the frequency of this phrase in u_1, u_2, and u_3 in Table 1 is 1, which validates the frequency constraint. Thirdly, it is the longest phrase started with "`eat`", thus the maximum constraint also holds. So this phrase is a high-quality phrase.

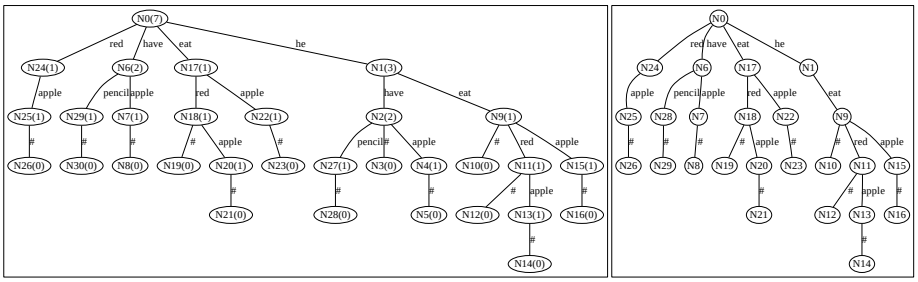
4.2 Generating High-Quality Phrases from A Sentence

Selecting Phrases with Gap and Maximum Constraints. As the high-quality phrases must satisfy gap constraint, we describe an algorithm to generate such phrases for a sentence. The aim of this model is extracting the sequential and syntactic relationships between words and modeling these relationships into a graph. A plain sentence model

(a sequence of words) cannot reach this point because we allow gaps in the phrases. Instead, we use the Minipar [8] to preprocess the sentence s and model it as a S-Graph, a directed acyclic graph, where each node represents a word in the sentence s, and there is an edge between two nodes, say s_i and s_j, if $j = i + 1$ or s_i has a dependency relationship with s_j. Obviously, the longest path started with one node in the graph corresponds to a phrase with gap and maximum constraints.

Selecting Phrases with Frequency Constraint. As the phrases must satisfy frequency constraint, we have to consider all the sentences in the database $\mathcal{U}$ to generate high-quality phrases of a sentence. We use a *frequency-trie* structure to collect frequencies of all possible high-quality phrases. We develop a two-step algorithm to achieve our goal as follows. In the first step, we use a frequency-trie to collect the frequencies of the phrases, and then we traverse the trie to prune frequent phrases.

Step1: Collecting the Phrase Frequencies: Each node in the frequency-trie has a number to represent the frequency of the phrase corresponding to the path from root node to this node (Note that we set the value in a leaf node as 0). Figure 2 (a) displays an example. The value 2 on node N2 denotes that the frequency of "`he have`" is 2.



(a) The trie for all possible phrases. (b) The trie for infrequent phrases.

Fig. 2. The frequency tries corresponding to u_1, u_2 and u_3 in Table 1.

We initialize the root node of the frequency-trie as 0. For each S-Graph $\mathcal{G}$ modeled from s, we collect all paths from the graph node n_i, which correspond to the phrases started with s_i. We examine whether these phrases are prefixes of the generated phrases. If not, they can be inserted into the phrase set P, and their prefixes in P which are previously generated should be removed. Thus the phrases in P satisfy both gap constraint and maximum constraint. Then, for each phrase in P, we insert it to the frequency-trie and increase the frequency for each node on the corresponding path. Next, for each node (excluding the root) on the path, we add a leaf node with an endmarker # to them if the leaf node does not exist to distinguish a phrase from its extended ones. Take u_2 in Table 1 as example. One longest phrase started with "`he`" is "`he eat apple`". It is inserted into the frequency-trie as shown in Figure 2 (a). The frequencies of nodes N1, N9 and N15 are increased and a new leaf node N10 is appended for the prefix "`he eat`".

Step2: Selecting Infrequent Phrases: The frequency-trie collects all the phrases satisfying the syntactic gap constraint and maximum constraint. In this step, we select the phrases based on the frequencies. The algorithm traverses the nodes of the frequency-trie in pre-order. For each node n with leaf node (which means there is a phrase corre-

sponding to the path from the root to n), if its frequency is not smaller than δ, we delete this node and recursively delete its children; otherwise, we recursively visit all of its children. According to this algorithm, given the threshold $\delta = 2$, the frequency-trie in Figure 2 (a) is converted to the trie in Figure 2 (b). For example, node N2 and its descendants are eliminated since N2's frequency is not smaller than δ. N1 and N17 should not be deleted, because they are infrequent or have no leaf node.

5 Experiments

In this section, we evaluate the performance of our proposed sentence matching method. We obtained two sets of parallel sentences from the Web to evaluate our proposed methods. ICIBA (http://www.iciba.com), denoted as D_I, contains 520,899 parallel sentences, most of which are expressions for everyday use. The average length of the English sentences is 13.2 words. CNKI (http://dict.cnki.net), denoted as D_C, contains 800,000 parallel sentences extracted from the academic papers. The average length of the sentences is 20.5 words. For each data set, we randomly selected 100 sentences as source sentences to be translated, and took the other sentences in the database $\mathcal{U}$.

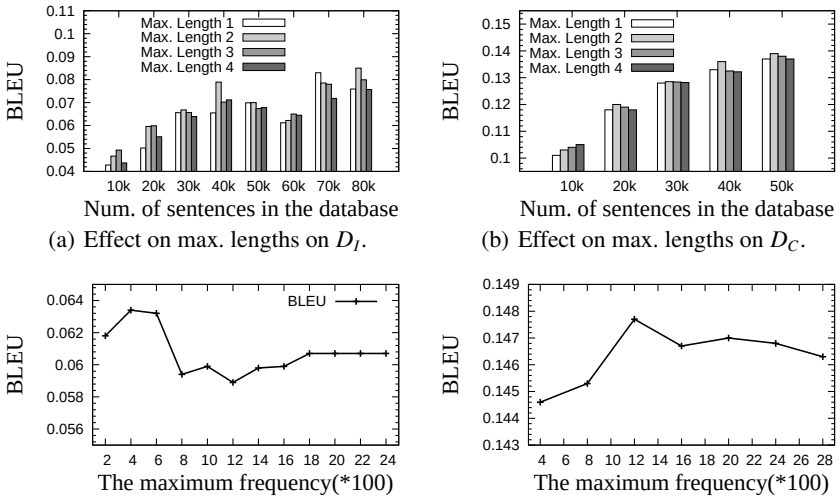
We compared our model with the three models on top of the selected phrase set: Jaccard coefficient, Cosine distance, and Edit distance. We also evaluated our proposed method with state-of-the-art sentence-matching methods. Translation Model based methods (TM) estimate the probability that one sentence is a translation of another by using a translation model, and take the probability as the similarity [10, 12]. We implemented the state-of-the-art translation model with its optimal parameters in [12]. Variable-length gram based methods (VGRAM) select variable-length grams, rather than fixed-length grams, as features of sentences [7]. We implemented the VGRAM and computed the cosine similarity with TF-IDF weights between gram sets.

We exploited the most popular metric *BLEU* in machine translation to evaluate the sentence-matching effectiveness. BLEU measures the similarity between the translation output and a reference translation using the N-gram strategy. Thus, it can measure the revision efforts of translators that transform the translation of the retrieved sentences to the reference translation. The BLEU score was calculated by the NIST script with default settings (http://www.nist.gov/speech/tests/mt/2008/scoring.html).

All the programs were implemented in JAVA and all the experiments were ran on a computer with an Intel Core 2 CPU 1.87 GHz, 2 GB RAM and 100 GB disk.

5.1 Effects of High-Quality Phrase Selection

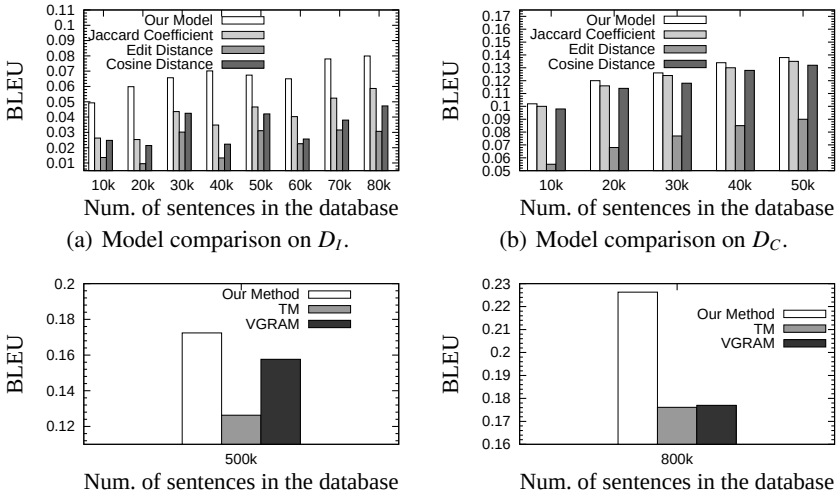
We first evaluated the effects of selecting phrases with different maximum length thresholds. Figures 3(a) and 3(b) illustrate the BLEU scores of phrases with different maximum length thresholds. We observe that selecting too long phrases does not yield too high performance. Surprisingly, the highest performance is achieved at the maximum length thresholds of 2 or 3. This can be explained by the maximum constraint introduced in Section 4.1. This constraint guarantees that each generated phrase is not a prefix of others. Therefore selecting too long phrases will eliminate many other possible high-quality ones, which will affect the performance. As the phrases must satisfy the frequency constraint, we also examined the effect of different maximum frequency



(a) Effect on max. lengths on D_I. (b) Effect on max. lengths on D_C.

(c) Effect on freq. thresholds on D_I. (d) Effect on freq. thresholds on D_C.

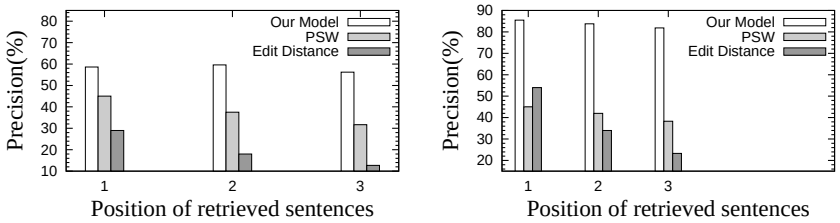
Fig. 3. Effect of Phrase Selection.



(a) Model comparison on D_I. (b) Model comparison on D_C.

(c) Approach comparison on D_I. (d) Approach comparison on D_C.

Fig. 4. Performance Comparison.



(a) Precision over 3 positions on D_I. (b) Precision over 3 positions on D_C

Fig. 5. The user study on the two data sets, D_I and D_C.

thresholds. Figures 3(c) and 3(d) give the experimental results on the two data sets. As the threshold increases, the performance first increases and then decreases at a fixed point (No phrase is eliminated given a large enough threshold). It indicates that selecting phrases within a maximum frequency indeed improves the performance of sentence matching. However, too many phrases will be eliminated if this threshold is too small.

5.2 Effectiveness Comparisons

Comparison of Similarity Models. We compared the effectiveness of Jaccard coefficient, edit distance, cosine distance, and our model (Section 3) on top of the selected phrase sets. Observed from Figures 4(a) and 4(b), our model achieves the highest BLEU score for various numbers of sentences in the database. This is because our model introduces the syntactic importance of each phrases by considering the matched words and unmatched words in the gaps, compared with Jaccard coefficient and cosine distance which only count the number (or weights) of shared phrases. The performance of edit distance is the worst. Because taking the minimum number of operations of transforming one sentence to the other as the similarity is not very useful for translation.

Comparison with Existing Matching Methods. We compared our proposed method with two baseline methods, TM and VGRAM. Figures 4(c) and 4(d) display their performance. Our method achieves the best BLEU scores and outperforms the baseline methods significantly. For example, the BLEU score of our method is 37% higher than that of TM in Figure 4(c), and is 30% higher than that of VGRAM in Figure 4(d). The improvement of our approach is mainly due to the selected high-performance phrases. TM assumes that each word translates by itself [12, 10] and neglects the order of words. VGRAM only considers continuous words, and misses some discontinuous phrases that reflect syntactic features of a sentences for translation. For example, consider the sentence, "he have eaten an red apple". The discontinuous words "he eat apple" is very important for translation. Unfortunately, they cannot be selected as features of the sentence by VGRAM. In contrast, our method allows discontinues words (i.e., phrases), and proposes effective rules to select high-quality phrases by considering syntactic and frequency information. The experimental results show that such information really plays an important role in sentence similarity measurement, and the selected phrases are more powerful in capturing the key features of sentences. Secondly, compared with the translation probability used by TM and the standard cosine model with the TF-IDF weight used by VGRAM, our proposed similarity model can further exploit the matched words and unmatched words in gaps in the selected phrases and then improve the sentence matching for translation.

User Studies We conducted user studies to evaluate the usefulness of our sentence matching method. We compared with the PSW (Percentage of Shared Words) and the edit distance, which are widely used in commercial STA systems (e.g., Youdao and Trados) and Translation Memory systems [13] respectively. We asked 10 volunteers to annotate whether each retrieved sentence translation is helpful for the translation of the source sentence. Then, we averaged their results and computed the *precision* according to these labels. Observed from Figures 5(a) and 5(b), our method outperforms the two baseline approaches significantly. For example, the precision at position 1 (i.e., $P@1$) of our method is nearly 50% higher than that of baseline methods in Figure 5(b). The

results show that our proposed methods are effective and helpful for assisting users to produce high-quality translations.

6 Conclusion

In this paper, we have proposed an effective approach for searching closest sentence translations from the Web. We introduced a phrase-based sentence similarity model and selected high-quality phrases from sentences for effectively quantifying the similarity between sentences. We have implemented our method and built an STA system, where parallel sentences are crawled and extracted from the Web. We have conducted extensive experiments to benchmark our method, and the experiment results show that our approach achieves high result quality, and outperforms state-of-the-art methods.

Acknowledgement

This work was supported by the National Natural Science Foundation of China under grant No. 61003004 and No. 60833003.

References

1. E. Biçici and M. Dymetman. Dynamic translation memory: Using statistical machine translation to improve translation memory fuzzy matches. In *CICLing*, pages 454–465, 2008.
2. K. Cai, J. Bu, C. Chen, and K. Liu. Exploration of term dependence in sentence retrieval. In *ACL*, 2007.
3. N. Cancedda, E. Gaussier, C. Goutte, and J. Renders. Word sequence kernels. *The Journal of Machine Learning Research*, 3:1059–1082, 2003.
4. F. Damerau. *Markov models and linguistic theory: an experimental study of a model for English*. Mouton De Gruyter, 1971.
5. I. Garcia. Power shifts in web-based translation memory. *Machine Translation*, 21(1):55–68, 2007.
6. H. Ichikawa, K. Hakoda, T. Hashimoto, and T. Tokunaga. Efficient sentence retrieval based on syntactic structure. In *ACL*, 2006.
7. C. Li, B. Wang, and X. Yang. Vgram: Improving performance of approximate queries on string collections using variable-length grams. In *VLDB*, pages 303–314, 2007.
8. D. Lin. DEPENDENCY-BASED EVALUATION OF MINIPAR. *Treebanks: Building and Using Parsed Corpora*, 2003.
9. I. Mel'cuk. *Dependency Syntax: Theory and Practice*. State University of New York Press, 1988.
10. D. Metzler, Y. Bernstein, W. B. Croft, A. Moffat, and J. Zobel. Similarity measures for tracking information flow. In *CIKM*, pages 517–524, 2005.
11. D. Metzler and W. B. Croft. A markov random field model for term dependencies. In *SIGIR*, pages 472–479, 2005.
12. V. Murdock and W. B. Croft. A translation model for sentence retrieval. In *HLT/EMNLP*, 2005.
13. E. Planas and O. Furuse. Multi-level similar segment matching algorithm for translation memories and example-based machine translation. In *COLING*, pages 621–627, 2000.