A Keyword-Based Method for Measuring Sentence Similarity

Yuanjun Bi, Kai Deng and JinXing Cheng
Suning R&D center, Palo Alto, CA, USA
{sophia.bi, kai.deng, jim.cheng}@ussuning.com

ABSTRACT

In this paper, a sentence similarity computing approach based on keywords is presented. First, it extracts the keywords from a sentence. Then the approach computes ranking scores for the keywords. Finally it applies these ranking scores into the sentence similarity computation using the Jaccard similarity coefficient. Experiments on a real word chatterbot system dataset demonstrate that this proposed approach significantly improves the relevance of sentence similarity method up to 30%.

1 INTRODUCTION

In recent years, chatterbot becomes a hot research area, which also has wide industrial applications. One important application is online customer services, which saves significant time and human resource by simply texting or asking a chatterbot. One common approach in a chatterbot system is to pull a response with the most matching sentence from a database, in which conversations are stored as question-answer pairs. Then answering a query can be reduced to the problem of similarity identification between the query and candidates in a database. Next, the chatterbot replies with the answer towards the most similar question. Most research work for detecting similarity between two documents focus on shared words [3]. But when it comes to a short term sentence, the similarity computation may be occupied with shared less informative words. Here we found that by using keywords to represent a sentence, we can avoid the confusion of noisy words. In this paper, we explored a novel method of computing ranking score for keywords and applying them to the similarity computation. We conducted the experiments on the Suning chatterbot system *Sunny* that provides online customer service for transactions on Suning.com. The task is to find the most similar question at the semantic level for a query.

2 KEYWORDS EXTRACTION AND RANKING

For computing the ranking score of keywords, we worked from three heuristic aspects. Firstly, we generated possible key terms from our database by their frequency. Secondly, we considered the parts of speech of the words. Last but not least, we took into account the position of the words. **(1) Keywords Extraction**. To generate the key terms from the database, we classified the database sentences by intents, such as greeting, shipping and returning. For each intent, we extracted top N frequent words. The reason for extracting top N frequent words from each class instead of from all documents is that, it is possible some classes have much less instances than other classes. Extracting keywords independently can avoid keywords in minor classes replaced by non keywords in major classes. For example, in our database, the greeting class has more queries than the shipping class. The word *"hello"* appears only two times, but it should be considered as a keyword in the greeting class. The word *"tomorrow"* appears five times, but it should not be considered as a keyword in the shipping class. Also, classification guarantees words that present the same intent will appear together [4]. After stop-words removal, we obtain the keywords list. **(2) Keywords Ranking**. For our database, we observed that verbs normally play an important role in a sentence. So in the ranking progress, we assign weight to words that are verbs. We also found that in our database, the intent of a query relies on the position of verb. Statically, the verb closer to the end of a sentence is more important than the verb in the front of a sentence. Therefore, we take into account the position for keywords ranking. The keywords ranking algorithm is shown in Algorithm 1.

Algorithm 1: Keywords Ranking Algorithm

Input: $Sentence \leftarrow \{w_1, w_2, ..., w_n\}$
$K \leftarrow$ generated keywords set
$decay_v \leftarrow$ verb decay weight
$decay_k \leftarrow$ non-verb decay weight
$score_v \leftarrow$ initialized verb score
$score_k \leftarrow$ initialized non-verb keywords score
$score_{min} \leftarrow$ bottom value of score
Output: keywords score s_i for each $w_i \in Sentence$;

```
1  S ← ∅
2  foreach w_i in Sentence in reverse order do
3      POS_i ← part-of-speech of w_i
4      if POS_i is verb then
5          s_i ← score_v
6          score_v ← max{score_min, score_v − decay_v}
7      else if w_i ∈ K then
8          s_i ← score_k
9          score_k ← max{score_min, score_k − decay_k}
10     else
11         s_i ← score_min
12     S ← S ∪ {(w_i, s_i)}
13 return S
```

The keywords set K in the input of Algorithm 1 is pre-generated from our database. This offline input may not work for words not in the database. To address this, line 4 assigns weights to verbs even though they are not shown in K. Besides the part-of-speech, we found that the words closer to the end of a sentence are more informative to represent its intent. To utilize this observation, we parsed words a sentence segments in reverse order. And decay weights in lines 6 and 9 guarantee scores become smaller as words approach the beginning of the sentence. Also, all the scores have a lower bound $score_{min}$ to avoid negative value.

WebSci '17, June 25-28, 2017, Troy, NY, USA
 978-1-4503-4896-6/17/06.
DOI: http://dx.doi.org/10.1145/3091478.3098878

Table 1: Differences evaluation compared to General

Category change	Keywords	Keywords&POS
0→ [1, 2]	54(28%)	75(31%)
2→ [1, 0]	44(23%)	53(22%)
0→0	93(49%)	113(47%)
Total	191	241

3 SENTENCE SIMILARITY

There is a large portion of work focused on the sentence similarity. Achananuparp [1] evaluate fourteen existing similarity score between two sentences. In this work, we choose Jaccard similarity coefficient to compute the similarity between two sentences. Jaccard similarity coefficient is defined as the size of intersection set divided by the size of the union set. i.e., Function 1 Here, Q is our query sentence words set. C is the candidate sentence words set. It is high efficient and easy to implement for industry application.

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} \tag{1}$$

To apply keywords acquired from Algorithm 1 into Function 1, we generated Function 2. In this function, the similarity score between query Q and candidate C equals the division of their intersection set score and union set score. Let $f(S, w_i)$ be the ranking score of the word w_i by applying Algorithm 1 on sentence S, where w_i is one word from sentence S. Then the intersection set score is the sum of $f(Q, w_i)$ for all words w_i both in Q and C. In the union set, if a word w_i only comes from query Q, then its ranking score is $f(Q, w_i)$. While a word w_i that only belongs to candidate C's gets the ranking score $f(C, w_i)$. Since the union score adds keywords score not in query Q, candidates whose keywords are different from keywords in query get a smaller similarity score. For example, *"Please ship my order."* has three intersection words with *"Please cancel my order."*. But since these two sentences' keywords(*"ship"* vs *"cancel"*) are different, it gets a smaller score from Function 2. This makes finding the matching candidate more reasonable. Function 2 reduces to Function 1 if all keywords scores are the same value.

$$J_{keywords}(Q,C) = \frac{\sum_{w_i \in Q \cap C} f(Q, w_i)}{\sum_{w_i \in Q} f(Q, w_i) + \sum_{w_j \in C \setminus Q} f(C, w_j)} \tag{2}$$

4 EXPERIMENTS

Experiment Setup: Our experiments are conducted on the database of Suning Chatbot system. There are 12, 816 question-answer pairs in our database. They can be classified into 22 different intents. By using top N frequent method, we generate 1, 827 keywords. We generated two kinds of testing data set. (1) **Raw Test** 2, 069 testing queries which are extracted from real-world human customer service online chatting logs. We use this dataset to evaluate our methods to questions that may not appear in the given database. (2) **Standard Test** 1, 724 testing queries which are all relevant to questions in our databases. For each query in testing cases, we used three methods to get the matching question. (1) **General** Jaccard similarity computation in Function 1; (2) **Keywords** method only considers words in the keywords file. That means line 4 to line 6 in Algorithm 1 will be ignored when computing the ranking score. Then use the score in Jaccard similarity computation of Function 2.

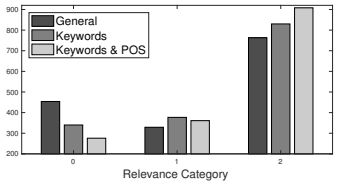

Figure 1: Distribution of relevance category on Raw Test

(3) **Keywords&POS** method applies ranking score from Algorithm 1 to the Jaccard similarity computation of Function 2.

Performance: We evaluated the effectiveness of three methods mentioned above at different levels of similarity. Define category 2 as *Exact Match*, 1 as *Relevant* and 0 as *Unrelated*. We first run three methods on Raw Test. Figure 1 shows the relevance category distribution, which clearly shows Keywords and Keywords&POS methods outperform the baseline General method. They have more cases in category 1 and 2, and less cases in category 0. In addition, Keywords&POS method performs much better than the Keywords method in category 2, which means Keywords&POS method has better similarity judgement when met with sentences with different representations. Then we run three methods on Standard Test. To compare the performance of Keywords and Keywords&POS methods in detail, we focused on cases where these three methods have different results. Table 1 shows that Keyword&POS method has more different cases than the Keywords method. This is because Keyword&POS method considers verbs that don't belong to key terms list. It changed the baseline results more aggressively. But Keyword&POS has a greater ratio change from Unrelated to more relevant(1 or 2), and has less ratio change from 2 or 1 to Unrelated. Both experiments indicate our keywords approach is more effective than original Jaccard similarity in measuring sentences' similarity.

5 CONCLUSIONS AND FUTURE WORK

We present a novel sentence similarity measurement which computes each words' ranking score utilizing words' frequency, part-of-speech and their position in the sentence. It applies theses scores on a modified Jaccard similarity coefficient method, to get the most matching candidate sentence. Experiments on a Suning Chatbot dataset validated the effectiveness of the proposed methods, and showed that the extracted keywords and words' part-of-speech significantly improved the overall effectiveness of finding the most similar sentence. For the future direction, instead of simply assigning larger weight to later position, we plan to study advanced utility of words' position [2] to further improve the performance.

REFERENCES

[1] Palakorn Achananuparp, Xiaohua Hu, and Xiajiong Shen. 2008. The evaluation of sentence similarity measures. In *International Conference on Data Warehousing and Knowledge Discovery*. Springer, 305–316.
[2] Yuhua Li, Zuhair Bandar, David McLean, and James O'shea. 2004. A Method for Measuring Sentence Similarity and iIts Application to Conversational Agents.. In *FLAIRS Conference*. 820–825.
[3] Donald Metzler, Yaniv Bernstein, W Bruce Croft, Alistair Moffat, and Justin Zobel. 2005. Similarity measures for tracking information flow. In *Proceedings of the 14th ACM international conference on Information and knowledge management*. ACM, 517–524.
[4] Rada Mihalcea, Courtney Corley, Carlo Strapparava, and others. 2006. Corpus-based and knowledge-based measures of text semantic similarity.