Jingwen Zhong

CSC447 Natural Language Processing - Final Project Report

University of Rochester

Instructor: Prof. James Allen

May. 14 2021

Aligning PropBank Sense and TRIPS Sense for Verbs

Introduction

Analyzing word senses between different knowledge resources is a recurring topic between people who study natural language processing. Since PropBank and TRIPS use a similar granularity for word senses, it is worth a look in analyzing the mapping between them. PropBank provides a practical word bank with over one million words to pull from and built-in predicate argument annotations (Loper& Palmer 2007). The Trips lexicon on the other end provides us with a smaller amount of words but contains more essential information beyond semantic information for our analysis such as syntactic template, semantic types from the ontology. (Bose, R& Allen, 2020). While PropBank is widely used, Bose (2020) discovered that PropBank is overwhelmingly monosemous and it has significantly fewer senses than TRIPS or WordNet. They also claimed that the biggest problem when comparing these resources is that PropBank does not have definable similarity metrics, unlike TRIPS and WordNet. Thus, this project is aiming to generate an algorithm to perform the mapping between PropBank sense and TRIPS senses for verbs.

Some Related studies have been done. Bakay et al. have compared the verb senses between PropBank and WordNet manually and found out most of the senses in PropBank have one-to-one correspondences in WordNet and, they found that their manually tagged method performs better than their automatically tagged resource (Bakay et al. 2019). Loper and Palmer created a lexical mapping between PropBank and VerbNet to get all the possible mappings of the senses, but they did not specify how they mapped them. They also created an instance classifier to choose the best result from the possible mappings by using WSD engine. (Loper& Palmer 2007).

This paper describes the strategy and rules for the mapping algorithm I create. The strategy mainly contains two parts, extracting information from PropBank and TRIPS and implementing the mapping algorithm. The rule for the mapping contains five steps, and after each step, the score of the aligning of each pairs will be updated. The final result is generated by the score of each pair.

Extracting Information

Extracting information is a very important part for the whole processing because choosing the right information and the right format can directly determine the accuracy of the mapping and the speed of the algorithm.

For PropBank, I extract information from the PropBank .xml files manually. The PropBank information I extract contains entry id, entry's definition, the WordNet sense keys that VerbNet provides, and examples. Take "abandon" as an example, the output of the function look like this:

```
1 find_propbank("abandon")
```

```
[['abandon.01',
  'leave behind',
  ['abandon%2:40:00::', 'abandon%2:38:00::'],
  ['and they believe the big board under mr phelan has abandoned their interest',
   'consumers power co  now the main unit of cms energy  ran into financial problems over its  42 bil
lion midland nuclear plant  which was abandoned as a nuclear facility in 1984 because of construction
delays and high costs ']],
 ['abandon.02', 'exchange', [], []],
 ['abandon.03',
  'surrender, give_over',
  [],
  ['once he had abandoned himself to the very worst once he had quieted all the dragons of worry and
suspense there would nt be very much for mae to do']]]
```

For TRIPS, I use Pytrips package to get the mapping from Trips to the lexicons that have either the input word or input word + preposition, and if none of them is found that means the type is directly mapped form the lexicon. Take “come” as an example, the entry should contain (ont::reviving, ont::start, ont::come, ont::become, and ont::occurring) and also ont::arrive because come%2:38: 04:: is a synonym of arrive%2:38:00. Then I put the entry id as the first element of the list.

The latter elements of the list are synsets of the entry, WordNet sense key, WordNet definitions, and the WordNet examples. The definitions and examples of the WordNet sense key that the entry related to are extracted from the collie-wnkey.json file. Take “abandon” as an example, the output of the function look like this:

```
1 find_trips("abandon")
```

```
[[ont::leave-behind,
  ['desert', 'desolate', 'ditch', 'forsake', 'leave', 'leave behind'],
  ['abandon%2:31:00::',
   'abandon%2:40:00::',
   'ditch%2:40:00::',
   'leave%2:30:03::',
   'leave%2:31:05::',
   'leave_behind%2:38:00::'],
  ["be survived by after one's death",
   'forsake, leave behind',
   'go and leave behind, either intentionally or by neglect or forgetfulness',
   'leave someone who needs or counts on you; leave in the lurch'],
  ['The mother deserted her children',
   'We abandoned the old car in the empty parking lot']]]
```

Implement Mapping Algorithm

In general, the mapping algorithm first will get all pairs of PropBank entries and TRIPS entries along with the mapping score as the candidates, and then decides the final output by the score.

Before implementing the algorithm, I conduct some functions so that I can change the format of the information for later mapping. First I create a function to find all the parents in the path of the TRIPS ontology, and a function of Wu Palmer metrics to get the similarity score. I also conduct a function to separate the sentence by punctuation. The result example of this function is as below, and the reason of changing sentences like this is that, later I change the definition of PropBank to this format, and I use Ngram to get any combination of the words for the definition sentence of TRIPS. Then I can compare if there are any Ngram combination of the words of TRIPS definition matches this separated PropBank definition.

```
1 split_sentences('come to: speaking of')
```

```
['come to', 'speaking of']
```

```
1 split_sentences('pursue (often with after), pursuing')
```

```
['pursue', 'often with after', 'pursuing']
```

To get the candidates, I outer loop all PropBank entries, inner loop all TRIPS sense. In each loop,

first I pass the PropBank entry with meaning of idiom, since I am assuming TRIPS doesn't have entry that explains idiom. Then I use four steps to calculate the score. The starting score is one. Step 1: Checking if the TRIP's entry matches the words in the entry definition of PropBank, if true, then score*2.; Step 2: Checking if any of the synsets in TRIPS match the words in the entry definition of PropBank, for every word in the synsets, if it matches then score*2.; Step 3: Checking if the WordNet sense keys in TRIPS match the WordNet sense keys in PropBank, if it matches, and the key's lemma name equals to the input word, then score*2. If keys match but do not equal to the input word, then count the number, then times score with the counted number. Step 4: Checking if there are any combination in the definition sentence of trips matching the entry definition in PropBank. The combination of words starts from length two, till the length of the sentence, here I use Ngrams function. I count how many combinations that matches the entry definition in PropBank, and then multiply score with this number. Step 5: the last steps is the similarity calculation. I use Wu Palmer metrics to get the similarity of the words in PropBank and the entry in TRIPS. Specifically, I only deal with the single word in PropBank not word + proposition or any phrase with more than one words. Then I find the highest similarity and multiply score with similarity.

After getting all the candidates and the mapping score, I can choose which candidate should be the final output. I create an empty checking list, so that I can append entry names of the final PropBank and TRIPS senses one by one, and later I can use this checking list to check whether the entry is already in the final output or not. Since the candidates are already sorted in descend order, so I always keep the first candidate in the final output, then append the entry names to the checking list. Then I delete candidates if 1. The score is smaller than the previous one and at least one of the entry name is in the checking list; 2. The score is less than 0.5; 3. Both entry names are in the checking list. If none of these matches, then kept it in the final output. The final output contains entry id for PropBank, entry id for TRIPS and the score.

The output of the mapping for the input file that contains "come" and "cover" looks like the figure below.

```
come.04  ont::become  2.2222222222222223
come.03  ont::occurring  1.0
come.02  ont::arrive  0.9
come.01  ont::come  0.8888888888888888


cover.02  ont::cause-cover  3.888888888888889
cover.01  ont::taking-care-of  1.4
cover.03  ont::protecting  0.7368421052631579
```

Conclusion and Future work

The output of the mapping between the senses of PropBank and TRIPS using the algorithm is not exactly accurate. Since the mapping for one concept to other concepts of the word subjects to opinions and the senses are based on context, it is hard to use the rigid rule to align them.

There are still problems in this algorithm that I can solve in the future. Take the above examples, first I notice that there is a discrepancy between Pytrips package and the web browser, the result contains "ont::arrive" and "ont::protecting" which are not showing on the web browser. Also, I don't know where to get the synsets words form lexicon lookup for TRIPS (i.e., hide words on the web browser), and it impacts a lot on the mapping. Another problem about the algorithm is that, since it is generated by the magnitude of the score, it is not able to align one kind of sense with multiple other

kind of senses unless the score is the same. For example, for word “cover”, “ont::cover” is missing here, but it should be aligned to cover.02 same as “ont::cause-cover”. Moreover, I don’t use the example sentences in both PropBank and TRIPS for the mapping. In the future I can generate a way to compare the example sentences such like comparing the sentence structure using some constituency parser or the subject/object types using TRIPS parser.

References:

Bakay, Ö., Avar, B. & Yıldız, O.T. (2020). Comparing sense categorization between English propbank and english wordnet. Paper presented at the Proceedings of the 10th Global WordNet Conference, 307-314.

Bose, Ritwik.University of Rochester, ProQuest Dissertations Publishing, 2020. 28091827. 100- 101.

Bose, R., Vashishtha, S., and Allen, J., “Hinting Semantic Parsing with Statistical Word Sense Disambiguation”, {\it arXiv e-prints}, arXiv:2006.15942.

Loper, E., Yi, S-t., & Palmer, M. (2007). Combining lexical resources: Mapping between PropBank and VerbNet. In: Proceedings of the 7th International Workshop on Computational Semantics. Tilburg, the Netherlands.