GRADUATE PROJECT
Aligning Word Senses and Roles between AMR and TRIPS

Rich Magnotti, Jeet Ketan Thaker, Ibrahim Khan, Jinhao Zhang
{rmagnott, jthaker, mkhan26, jzh160}@ur.rochester.edu

1 Introduction

Given two distinct English semantic representations, AMR and TRIPS ontology, our task was to attempt to find a mapping of word senses and roles from AMR to TRIPS and automate it. This was possible because of the similar granularity of word senses between the two representations. Our contributions include a preliminary functional system to take as input some words and then a) match each PropBank roleset with a TRIPS ontology word sense, then once senses are matched, b) match each role between the AMR word sense and the roles in the candidate TRIPS type word sense. Our system accomplishes this by the following criteria: (i) iterate through each word sense belonging to AMR and tracks all candidate word-sense matches in the TRIPS ontology, (ii) rank and prune the relations between input AMR word-sense and candidate TRIPS match, (iii) output its closest word senses match provided by TRIPS, (iv) repeat (i)-(iii) for each word sense, and once again for each role when appropriate.

2 Methods

2.1 Sense Matching

The criteria by which word-senses are able to be matched from PropBank are: syntactic function, roleset definition, FrameNet alias, verbnet alias, and sample sentences provided by PropBank. The TRIPS matching criteria possible to utilize is: ontology word senses, type synset, words belonging to the ontology type. We performed a number of tests in order to empirically determine which criterion is the most relevant to getting "good" matches. Ultimately *syntactic function* and *definition similarity* were chosen, as the other criteria either proved not to be useful or not possible to extract. This matching algorithm is performed exhaustively, i.e. given some word, go through each AMR roleset and find the best matching type from TRIPS (or no match) until there are no more AMR rolesets.

As a primary means of matching, the syntactic function of the word is considered. Although AMR exclusively contains verbs, we still ascertain the syntactic function in order to then consider only that syntactic function in TRIPS. As a secondary means of matching, the PropBank definition is compared with the TRIPS type synset. Despite PropBank and TRIPS being similar in execution, the comparison is not binary compatible. As a consequence, it is necessary to perform definition matches based on *closeness* and not exactness. Therefore, the algorithm incorporates the use of the Python *spaCy* package (see appendix for setup instructions and usage) to perform the measure of closeness of definition to synset. Specifically, for each word or phrase in the PropBank roleset definition, all synonyms are

determined, and a comparison is made with all synonyms for each word in the TRIPS type synset. Only once these $O(n^2)$ comparisons are made will a score be assigned. Once all possible scores for each TRIPS candidate match are calculated, the highest will be chosen as the most likely match.

The algorithm goes as follows:

- Propbank XML file for the query word is parsed to extract syntactic function, definition and examples for each roleset.
- Definitions are parsed for individual semantic units. This is done by slicing the definition by commas to extract phrases or words that are semantically close to the roleset.
- For each extracted word or phrase, synsets from WordNet are extracted and stored. For 'm' rolesets, we'll have 'm' sets of words that closely associate in meaning to the roleset.
- We extract all ontologies from TRIPS corresponding to the query word. We pre-process all the synsets obtained.
- For each ontology, we extract synsets. For 'n' ontologies we'll have 'n' sets of words that closely associate in meaning to the ontology.
- Synset of each of the 'm' rolesets is compared with the synset of each of the 'n' ontologies and similarity scores are computed. The similarity score between a roleset and an ontology is the average of similarity scores of combination of all words or phrases in their synsets. We'll get O(m*n) similarity scores.
- The word or phrase similarity scores are computed using SpaCy but we apply a threshold on the similarity determined by the library. Words that are too far apart and have very low similarity are discarded and their similarity score converted to zero.
- threshold_function(x) = x if x > threshold else 0
- Words that match fully by simple string comparison carry a similarity of 1 and automatically qualify the equivalence between the roleset and ontology.
- If automatic equivalence as above is not obtained, for every roleset, the ontology with which it has the greatest similarity score is selected.

2.2 Role Matching

The main task for each sense matching pair is to match each of these roles between PropBank roles and TRIPS roles.
First, we extract information which will help us for role matching which is:

- What all roles does the PropBank sense have. Along with this information we also extract the definition of these roles and the label corresponding to each role. Labels are something like GOL or LOC etc,
- Next, we extract verbnet roles for each of the ARG's if they are present,
- The third piece of information is all the roles that TRIPS ontology can take along with optionality. It is something like, AGENT - REQUIRED, FORMAL - OPTIONAL etc.

First, we match ARG0:

We read the documentation and ARG0 is given to the causer or the experiencer of the event so it's matching is simple. If in the TRIPS roles list we find EXPERIENCER then we match ARG0 to EXPERIENCER. If the TRIPS role has AGENT we match ARG0 to AGENT.

To match ARG1, we first check if the definition of ARG1 contains the word 'thing'. Since the 'thing' is always combined with a word in a negative form, such as 'thing organized'.
If it does then ARG1 is matched to AFFECTED
If not, we check if ARG1 has associated verbnet roles.
If it does :

Then we have created a dictionary that maps verbnet roles to TRIPS roles. We get these extracted TRIPS roles from the verbnet role and intersect with the TRIPS roles that we got from the trips ontology and match ARG1 with the intersection.

We take care that the intersection does not contain experiencer or agent role as that is reserved for ARG0 , we also make sure that if ARG0 is experiencer then ARG1 cannot be AFFECTED.

Along with matching ARG1 with this intersection we also print out the optionality of roles, and roles that have the optionality REQUIRED will be considered final.

The importance order is REQUIRED >ESSENTIAL > OPTIONAL.

If no verbnet roles are present, then we default ARG1 to match to AFFECTED.

The algorithm for matching ARG2, 3, and 4 is the same which is as follows

If verbnet roles are present:

A = set of TRIPS roles gotten from labels (GOL etc) we made a mapping between labels and TRIPS roles,

B = set of TRIPS roles gotten from verbnet as we made the dictionary mapping verbnet to trips,

C is the intersection of A and B,

If C is not empty then ARG2 is matched with C,

If C is empty then,

D = set of roles from trips ontology,

ARG2 is matched with the intersection of D and A

If there are no verbnet roles available then :

A = same as what was A above,

D = same as what was D above,

ARG2 matched to the intersection of A and D

2.3 Tests and Grading

The grading criteria for a match involves a weighting based on number correct matches and normalization - and we determine the gold matches by hand. For example, if a target AMR word definition has synonyms that match 7 out of 7 synonyms of the words in the candidate match, we assign that a full score of 1 for sense-matching. If for example, it were to only match 5 out of 7, we would assign that a 5/7 or 0.71. Because we were not able to combine the algorithms for 4.3, it was decided to keep the grading for sense-matching and

role-matching separate. However, the same spirit of grading applies to role matching - the number of roles matched divided by the total number of roles able to match. The samples were chosen by hand at random, falling into three categories of complexity - simple: roughly one roleset matching to a couple of candidate types or vice versa, medium: roughly a couple rolesets matching to a couple of candidate types, and complex: many rolesets matching to many candidate types.

3 Discussion

3.1 Comments

The algorithms are successfully able to match sense and roles with relatively high accuracy - and can be found in the appendix under each respective example word. However, recall that the gold standard for scoring is based on by-hand solutions, which could lead to artificially high or low scores, depending on how reliably the individual hand-scored the example. Particularly, our algorithms scored reliably well on simple and medium examples, and surprisingly well on more complex examples. Across all examples, our algorithms scored:

Senses: 81.5%
Roles: 81.95%

It would have been ideal to also have made comparisons between AMR and TRIPS example sentences as a tertiary means of sense-matching. However, two reasons we did not select these criteria are that TRIPS does not always supply an example for its sense, and more importantly, we could not find a way to reliably extract TRIPS examples. TRIPS type examples are not provided in *pytrips* or *jsontrips* to our knowledge. Some preliminary efforts proved that although it is possible to extract TRIPS examples using *Selenium/webdriver* framework, but, this was haphazard and not always reliable. Therefore, we reluctantly chose not to utilize example sentences in our criteria.
As a criticism, some issues of code compatibility arose once the subteams had developed their respective algorithms. Although there were lengthy discussions as to the algorithms' overall designs, there were ultimately some data structure incompatibilities - and unfortunately as a consequence, we were not able to merge the two algorithms as initially planned for graduate proposal idea 4.3.

4 Conclusion

To reiterate the goal of the project, (4.1) given some user input word, develop some algorithm to match each PropBank roleset to the best TRIPS ontology word-type sense. Additionally, once a word sense match is found, (4.2) match each PropBank role to each role in the TRIPS word-type. We implemented the solution to these tasks by utilizing two distinct algorithms and straightforward grading criteria, in order to determine which word-senses and roles are the best matches respectively. We hand-scored 18 examples,

and compared our algorithm's results with our gold standard, providing relatively high output accuracy.

A Appendix

A.1 *spaCy*

Comments on spaCy:
spaCy utilizes a database of word-vector representations and performs cosine similarity comparison to assign a similarity score. More information can be found on the spaCy website https://spacy.io/usage/vectors-similarity.

NOTE: We found empirically that a score below 0.1 typically represents a poor similarity. However, a score above 0.3 typically represents an increasingly high similarity.

Usage and Installation:
The spaCy word-vector database can be downloaded via terminal command with

```
!python -m spacy download en_core_web_lg
```

spaCy can then be imported and the database loaded with

```
import spacy
nlp = spacy.load("en_core_web_lg")
```

Then, load the target comparison words as

```
t1 = nlp('example word')
t2 = nlp('example word')
```

Finally, the comparison can be made with

```
print(t1.similarity(t2))
```

Which outputs a score between 0 and 1.

A.2 Samples

Simple:
01 Eat
eat.01 and EAT
Arg0: Agent
Arg1 : Affected

Sense score: 1
Role score: 1

02 Climb
climb.01 and LOCOMTE-UP
arg0: climber->AGENT
arg1: things climbed->AFFECTED

Sense Score : 1
Roles Score : 1

03 Admire
admire.01 and APPRECIATE
arg0: admirer->EXPERIENCER
arg1: admired->NEUTRAL or FORMAL

Sense score: 1
Role score: 1

04 Delay
delay.01 = ONT::delay
Arg0 = agent
Arg1 = affected
Arg2 = formal

Sense Score : 1
Roles Score : 0.67

05 Hook
hook.01 = ONT::attach
Arg0 = agent1
Arg1 = affected
Arg2 = affected1
Arg3 = result

Sense score: 1
Role score: 0.75

06 Place
place.01 = ONT::put
Arg0 = agent
Arg1 = affected
Arg2 = result

Sense Score : 0
Roles Score : 1

07 Park
park.01 = ONT::place-in-position
Arg0 = agent
Arg1 = affected
Arg2 = result

Sense score: 1
Role score: 1

Moderate:
08 Tell
tell.01 = ONT::tell
Arg0 = agent
Arg1 = formal
Arg2 = neutral

Sense Score : 1
Roles Score : 0.33

09 Reach
reach.01 = ONT::reach
Arg0 = agent
Arg1 = result

Sense score: 1
Role score: 0.5

10 Relax
relax.01 = ONT::loosen
Arg0 = agent
Arg1 = affected

Sense Score : 0
Roles Score : 1

11 Wear
wear.01 = ONT::wear
Arg0 = agent
Arg1 = affected

Sense score: 1
Role score: 1

12 Sing
sing.01 and SING

arg0: singer ->AGENT
arg1: song->FORMAL
arg2: audience->BENEFICIARY

Sense Score : 1
Roles Score : 1

13 Cluster
cluster.01 and JOINING
arg0: clusterer->AGENT
arg1: things clustered->AFFECTED
arg2: explicit mention of cluster->LOCATION or RESULT

Sense score: 1
Role Score: 1

14 See
see.01 and ACTIVE_PERCEPTION
arg0: viewer->EXPERIENCER
arg1: thing viewed->NEUTRAL
arg2: attribute of arg1, further description->unmatched

Sense Score : 0
Roles Score : 0.66

Complex:
15 Get
get.01 = ONT::acquire
Arg0 = affected
Arg1 = not matched or result
Arg2 = agent
Arg3 = not matched
Arg4 = affected-result

get.02 = not matched

get.03 = ONT::become
Arg0 = affected
Arg1 = formal

get.04 = ONT::cause-effect or ONT::make-it-so
Arg0 = agent
Arg1 = result

get.05 = ONT::passive
Arg0 = agent

Arg1 = affected
Arg2 = result

get.06 = ONT::make-it-so
Arg0 = agent
Arg1 = formal

get.22 = no match

get.24 = could fit any b/c so vague

get.28 = no match

get.30 = ONT::come-to-understand
Arg0 = agent
Arg1 = formal
Arg2 = source

get_out.01 = no match

Sense score: 0.67
Role score: 0.3616

16 Earn
earn.01 and EARNING
arg0: earner->AGENT
arg1: wages->NEUTRAL
arg2: benefactive->BENEFICIARY
arg3: source->SOURCE

Sense Score : 1
Roles Score : 1

17 Authorize
authorize.01 and APPROVE-AUTHORIZE
arg0: allower->AGENT
arg1: action allowed->FORMAL
arg2: explicit allowed agent->unmatched
arg3: purpose-> REASON

Sense score: 1
Role score: 0.67

18 Organize
organize.01 and ARRANGING
arg0: organizer -> AGENT

arg1: things organized ->AFFECTED
arg2: previous state -> SOURCE
arg3: benefactive -> BENEFICIARY
arg4: end state ->RESULT

Sense Score : 1
Roles Score : 0.8

Total Sense Score : 14.67/18 = 81.5%
Total Roles Score : 14.7516/18 = 81.95%