**Overview**

We participated in the 2013 TAC-KBP evaluation with a system that applied rules to Open IE tuples to find extractions for the set of KBP relations. These include per:origin (a person’s country or nationality) and per:title (a person’s job title). Our Open IE system has fairly high recall when a relation is expressed with a verb (e.g. “George Green was hired as president of Acme Widgets”) and limited recall for relations found within noun phrases (NPs) (e.g. “Acme Widgets president George Green stated …”).

Unfortunately, Open IE can only extract information when the relation phrase is explicit in the sentence. In the previous examples, the tuples are (George Green, was hired as, president of Acme Widgets) and (George Green, [be] president [of], Acme Widgets).

We would like to extend Open IE to extract from NPs where there is no explicit relation phrase. For example:

“French journalist Jean LeGall reported …”

(Jean LeGall, [has nationality], French)

(Jean LeGall, [has jobTitle], journalist)

In this example the implicit relation “has nationality” comes from the semantic type of “French” and the implicit relation “has jobTitle” comes from the semantic type of “journalist”. This pre-supposes that a set of task-specific semantic taggers have been applied to the sentence.

**Inputs**

The inputs and outputs will be saved in /projects/WebWare6/ImplicitRelationExtractor

We have lists of terms for the following classes from CMU’s NELL project:

jobTitle

nationality

religion

city

province

country

The text corpus used in the 2013 TAC-KBP evaluation is found in

/projects/WebWare6/DEFT corpora/TAC\_2010/TAC\_2010\_KBP\_Source\_Data/data/2009

/projects/WebWare6/DEFT corpora/TAC\_2010/TAC\_2010\_KBP\_Source\_Data/data/2010

Create a small development set with about 5 sentences each that have an implicit [has jobTitle], [has nationality], and [has religion]. (We can ignore the relations involving city, province, or country for now). You can find example sentences by grepping in the TAC-KBP corpus for particular job titles, nationalities, and religions. Be sure to include test sentences that have multi-word terms.

Later, we will do evaluation on a larger development set of sentence, and finally a formal evaluation on extractions over the entire set of TAC-KBP documents that contain a mention of one of the 2013 query entities. That test corpus will be available by then from another project, the Information Omnivore.

**Preprocessing and semantic tagging**

Build a system that hold lists of terms *t* for a set of classes *c* in memory, and takes a sentence *s* as input.

The system output is zero or more tuples for the sentence that have the following:

(Arg1, Rel, Arg2), *s*

where Arg1 is a noun phrase from *s*, Rel is [has *c*], and Arg2 is a noun phrase from *s* of type c.

Processing:

For each class *c*, find any phrase *p* from *s* that is found as a term in *c*, looking for the longest matching term.

If *s* has at least one match for a class *c*, run the Stanford parser on the sentence.

Examine the dependency parse to see if *p* is a modifier of a noun phrase *n*.

Note that the parse will only have a link from the head word of *p* to the head word of *n*.

You will need to expand *n* to get the entire noun phrase.

If *p* is a modifier of *n*, output the tuple (*n*, [has *c*], *p*), *s*.

You will need to code a keyword tagger that takes a list of terms and a class name as input. It walks through the words in a sentence looking for the longest subsequence of terms that are found in the list. You can assume that terms for a class have no more than *k* words, with a small *k* such as 3 or 4. You may want to create a hash table for possible prefixes of multi-word terms in the actual hash table of terms for the class.

You will also need to write code that calls the Stanford parser. Our programmer, Natalie, or any of the grad students, Xiao, Congle, or Angli, can help you with this. The most robust rules will be based on dependency parse links that indicate a noun modifier. Other rules could be simply based on the sequence of words in the sentence, as long as you verify that *n* is a noun phrase.

**Evaluation**

On your small development set, you can examine the output to see that there is a tuple for each sentence that has an implicit relation. This is debugging rather than a formal evaluation.

The first evaluation is to take as a test set, a fairly large set of sentences (~ 50K) drawn from the KBP corpus. For now, let’s take sentences as input rather than entire documents. Natalie can help you create a test set of sentences. Most of these sentences won’t have an implicit relation if they are drawn at random from the corpus.

Apply your system to those sentences to create a set of output tuples. If you have more than 1K tuples, randomly select about 1K to manually tag for precision. If you get less than 1K, start with a larger test set. Tag the tuples as correct (1) or error (0), where correct means that arg1 does have the implicit relation to arg2, based on the sentence. The percentage of correct tuples gives you the precision of your system. If you run another version of your system on the same test set, most of the tuples will already be tagged. You need to tag any new tuples.

Recall is harder to measure. This involves taking a sample of the sentences, not a sample of the tuples. Look through this sample to see if there are any implicit jobTitles, nationalities, or religions. This gives you an exhaustive answer key of the number of possible extractions for those sentences. Run your system to see what fraction of those extractions it finds. This is the recall.

Later, we need to do a formal evaluation on sentences that are relevant to the TAC-KBP query entities. We’ll only do this after the system has been informally evaluated as described above.

**Note:**

If any part of this design spec seems wrong or incomplete, it probably is. This is off the top of my head before actually trying any of this out.