Further Design for Implicit Relation Extractor **December 29, 2014**
S. Soderland

This is an extension of ImplicitRelation-designDoc.docx and DevSetPatterns.docx in WebWare6\ImplicitRelationExtractor\Documentation.

What we have now

1. Keyword class tagger with keyword list for origin, religion, and jobTitle
2. A function that operates on a list of sentences to do
    a. tag terms with keyword class tagger
    b. dependency parsing and NP-chunking
    c. apply rules to find entity modified by term
    d. identify NP boundaries of entity
    e. output a triple (entity, class, term)
3. A scoring / trace function that consults a set of expected extractions for the 19 sentence small dev set

Modifications to the extractor

Each implicit relation can have an entity types associated with it. The relations "has origin", "has religion", and "has jobTitle" expect the entity to be Person. Apply the Stanford NER tagger to the sentence as well as the parser. Modify the rules to require that the word for the entity be within an NER tag of the appropriate type.

The rule that looks for *conj_and(n1, n2)* as a mis-parse for *appos(n1, n2)* can be refined to only apply when there is no "and" between n1 and n2. It will still overgeneralize but not as badly.

If we have time, our approach of tagging terms from keyword lists can be used to extract other KBP relations from noun phrases. These may require different rules for different class lists. Let's get origin, religion, and jobTitle working before we tackle anything further.

Evaluating on the 1K dev set

For testing the implicit relation extractor on 1000 random TAC files in WebWare6\ImplicitRelationExtractor\ ImplicitRelationExtractor\tac_development\random-1000\sources.

This takes a two step approach. First the we extract all of the sentences from the sources and place it in one file in the sentences folder. Then we run the extractor on the sentence file and store it in a file in the results folder.

The sentence and results file has the same name is prefixed with a sequence number that is stored in the file name "file_sequence_number". The sentence file is structured so that each line has the sentence sequence number, source filename, and then the sentence, each separated by a tab. The results file is structured the same way, but replaces the sentence with the extraction result, which should include the sentence.

We will run the extractor multiple times as it is debugged and refined. So there will be multiple results folders with names that indicate a run date or other suffix.

Once we have an initial set of results, we will create an answer key from a random set of sentences. Save the random set of sentences so we can use the same ones each time as the system is improved

Tab-delimited fields in the results file:

DocID
SentenceID
Entity // the NP that has the extraction, e.g. "President George W. Bush"

Relation	// the class name, e.g. jobTitle
Slotfill	// the term with that class, e.g. "President"
Sentence	// text of the sentence
Tag	// "Correct", "Error" based on the answer key or empty if not found in answer key

The first time we run the system, there will be no answer key. We will tag each row of this answer key with 1 for correct or 0 for error. This becomes the initial answer key.

Write a scoring program that takes as input

- a set of results
- an answer key.

Outputs are

- a score report that has "Correct", "Error", or empty tag based on the answer key, and has summary statistics at the end with number correct, number of errors, and precision
- a file that lists only the extractions that are not in the answer key. After we tag these, they will be added to the answer key. We are likely to have a few new extractions from every modification of the system.

We may end up with alternate versions of an extraction as separate rows in the answer key if different systems give different NP boundaries. This should not cause any problem.

There can be a program flag in the scoring program that sets on verbose trace for particular sentences to help debug the system.

Formal KBP evaluation

The KBP Slot filling evaluation has a set of query entities and a set of target relations. We can evaluate on queries from 2013 and 2014. Each year there are 50 entities of type Person and 50 of type Organization. We will just use the 50 Person entities, and only evaluate on the relations

- per:origin
- per:religion
- per:title

Natalie can help us generate a set of relevant documents from the KBP corpus that a contain a mention of a query entity. We can generate sentences for those documents and further filter it to have only sentences with at least a partial match to the query entity. So, if the query entity is "Paul Gray", keep only sentences that have either "Paul" or "Gray", case sensitive match. This gives us a sentence file for each query entity.

Run the extractor on each sentence and keep only extractions where the extracted entity contains the query entity (one version of the system uses the full entity name, another does partial match). The version with partial match is an approximation of using coreference resolution to expand recall.

We also have the official results of using OpenIE-KBP on the 2013 and 2014 queries. This gives us three data points for our main ImplIE extractor

1. Implicit relation extractions: number correct, and precision
2. OpenIE-KBP extractions: number correct, and precision
3. OpenIE-KBP extractions plus Implicit relation extractions: number correct, and precision

Beside the main experiment, we should do ablation studies where we turn off some functionality of the system. For example, compare all extractions including partial match to query entity with strict match. Compare the full set of rules to only the most reliable rules. Perhaps other ablation experiments.