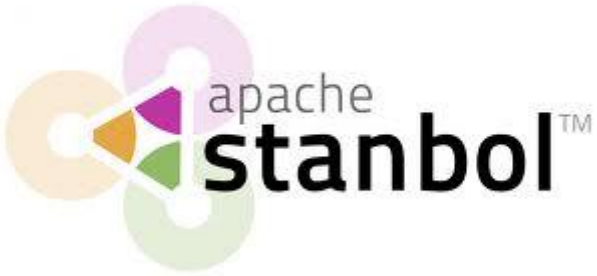
Google Summer of Code



Enahncement Engine for Entity Disambiguation

MID TERM REPORT

By: Kritarth Anand

Sup: Rupert Westenthaler

Introduction

Entity disambiguation is challenging problem to be addressed in Stanbol. Consider the example where the text to be enhanced is

"Paris is a small city in the state of Texas."

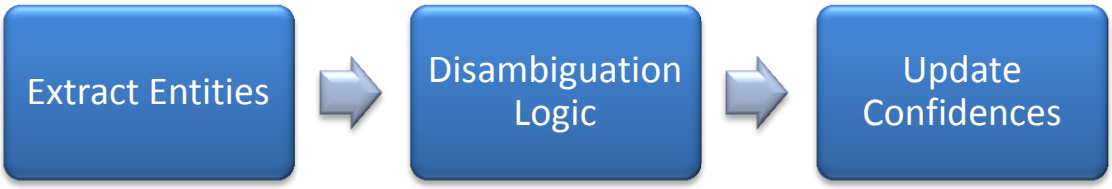
The Entities detected is these statements are

Paris	**Texas**
L1: Paris	L1: Texas, US
L2: Paris, Texas	L2: Austin, Texas
	L3: Denton, Texas

The objective of this enhancement engine is to understand disambiguate between the entities and map to correct Paris and correct Texas Entities. This can be done by taking into consideration the entire text present and analyzing it to disambiguate entities.

Approach

The EntityDisambiguation engine is expected to perform just ebfore the final linking. Yhe behavior can be explained as follows:



- **Extract Entities**: For all the Text annotations in the given text we need to provide all the entities that can be potentially linked to it. These have been calculated by the previous engines and hence are the part of ContentItem inputted. For each text annotation there exist many possible entitites and the prime goal of this entire exercise is to disambiguate between them.
- **Disambiguation Logic**: When an array of Map of text annotation to list of Entites (ambiguities) and the text itself is passed into this particular logic. It analyzes the parameters and then provides estimations of entities as to which entity is a better candidate for being mapped to the annotation.
- **Updating Confidences:** In this part the result of the disambiguation logic is written in the metadata by updating confidences of all ambiguities and there by successfully disambiguating the entities.

Disambiguation Algorithms

The performance of the engine is totally determined by the algorithm it uses for disambiguation.

Algorithm 1: A very trivial algorithm for disambiguation can be keep a track of all entities detected in the text and/or the selection context and then pass this information for a MoreLikeThis Query to Dbpedia. The results returned can be used to rank links and thereby disambiguate.

Though the algorithm seems intuitive it does not perform really well considering the fact that the fact that all annotations are given equal importance. So it might be the case the most relevant candidate gets subdued. The bottom line is that we would need better algorithms to match or define similarity.

Similarity defined by Neighborhood comparison:

The approach draws inspiration from social networks. Suppose we are provided with an entity A (their dbPedia or Wikipedia pages), now its page will have many link to certain entities. All these entities are called the neighborhood of A.

This paper describes a way of using neighborhood comparison for entity disambiguation. Jaccards's coefficient of similarity between two entities is the ratio intersection of neighborhood to union of neighborhood. This is a good detector of similarity as it if two entities are quite similar a large number of their common links to total links ratio.

The Jaccard's coefficient or Adamic/Adar similarity can be used as a good measure to rank entities for disambiguation.

Spotlight Approach:

The DBpedia addresses the same problem of entity disambiguation for DBpedia. This approach would involve the use of entire text inputted for disambiguation. The Spotlight approach would involve modeling DBpedia resource occurrence in a Vector Space model, each point is multi-dimensional space of words. The words have weights assigned which are defined as the product of TF and ICF. TF(Term frequency) represents relevance of a word for a given text(frequency). The intuition behind ICF is that the discriminative power of a word is inversely proportional to the number of DBpedia resources it is associated with. Let R_s be the set of candidate resources for a text annotation s. Let $n(w_j)$ be the total number of resources in R_s that are associated with the word w_j.

$$ICF(w_j) = \log \frac{|R_s|}{n(w_j)} = \log |R_s| - \log n(w_j)$$

Disambiguation is carried out by ranking possible entities(or different ambiguities of entities) according to the similarity score between their context vectors and the context surrounding the annotation or (the entire text inputted). For more details refer here. I am not sure if we can play around that much with any vocabulary and not just DBpedia.

Wikify Approach:

Wikify carries out entity disambiguation over Wikipedia. The Wikify! paper takes a step forward they use combination of the classic Lesk's Algorithm style approach(which is essentially looking for disambiguation from surrounding text) and also data-driven approach.

The second approach integrates the local and topical feature into a simple Machine Learning classifier (Bayes). Feature vectors are to be extracted from each possible entity of a particular text annotation. To model the feature vector certain surrounding words and parts of speech , global context (determined by certain sense specific words deteremined by certain parameters). The features are integrated in a Naive Bayes classifier.

Now they also carried out a mechanism to handle disperancy between the results outputted by the two approaches they use. The disagreement is result between two approach would mean potential chances for errors.