Named Entity Recognition and Wikipedia

Jake Brawer

May 15, 2016

1 Introduction

Wikipedia is an incredibly useful tool for quickly accessing knowledge about almost any topic. However, surprisingly, Wikipedia is also an excellent resource for natural language processing (NLP). This is due in part to the fairly predictable structure of Wikipedia articles paired with the inherent web of interrelations endemic to each article in the form of hyperlinks. In terms of named entity recognition (NER), Wikipedia may prove beneficial due to the fact that named entities for a given article are usually the very hyperlinks for that article. This means that articles are preannotated for named entities. While these links are never classified by type, this information is usually readily available in the links' corresponding page, often in the first sentence. The goal of this project is to create an algorithm that can classify all the hyperlinks on a given Wikipedia page. The benefit of such an algorithm is that it could create a near limitless corpus of structured text with annotated and classified named entities. Alternatively, such an algorithm could be used to classify identified named entities in other texts, assuming they have a corresponding Wikipedia article, making Wikipedia essentially a gigantic gazetteer.

2 The Data

All data was scrapped directly from Wikipedia itself. Wikipedia is an online encyclopedia with over 5 millions English articles. With few exceptions, the first sentence of any given Wikipedia article contains information germane to the task of classfication. Take these two sentences for example:

> "Barack Obama is an American **politician** serving as the 44th President of the United States."

> "Vassar College is a private, coeducational, liberal arts **college** in the town of Poughkeepsie, New York, in the United States."

Note that the words in bold provide clues for possible classifications for their respective articles (PERSON and INSTITUTION, respectively). Therefore I only try to classify articles based on the first sentence of each article in question.

3 Annotation Scheme

According to to Chapter 7 of the NLTK textbook, named entities generally belong to one of nine classifications: ORGANIZATION, PERSON, LOCATION, DATE, TIME, MONEY PERCENT, FACILITY, and GPE (geo-political entity). For the purposes of this project I have removed GPE and PERCENT as possible classifications. GPE was removed because it is a multiword category, which was unwieldy in my classification scheme (see methodology section). PERCENT was removed because few Wikipedia articles are about percents, or have a percent in the title. However, given the diversity of article topics, I added the following categories: FEATURE, CONCEPT, TOOL, VEHICLE, ANIMAL, PLANT, STRUCTURE. For a given Wikipedia page, the algorithm described here will attempt to categorize each hyperlink into one of these categories.

4 Methodology

In order to scrape Wikipedia articles, I employed the python module `wikipedia`. This module treats each page as an object, with things like the page's content and links as separate, easily accessible attributes. Information extraction was generally accomplished through NLTK functions and regular expressions.

The algorithm begins by analyzing at the first sentence in the corresponding article for each hyperlink in the article in question. The sentence is first preprocessed, with potentially confusing text like pronunciation information and parentheticals removed. The algorithm then looks for a third-person verb like is, was, are, were, etc., which likely denotes that an explanatory noun is forthcoming (e.g. "Barack Obama *is* an American **politician**"). If a verb is found, the remainder of the sentence is tokenized and tagged using NLTK. If one of the aformentioned verbs is not found, then the algorithm assumes that the most explanatory noun can be found in the title of the of article itself (e.g. "1900 United States **Census**"), and instead tokenizes and tags that.

This tagged sentence fragment is then passed into regex parser implementing a grammar of my own design. The grammar itself attempts to isolate the part of the sentence containing the most explanatory noun. It was incredibly difficult designing this grammar. Though Wikipedia articles stylistically tend to be pretty uniform,there is enough variation in the way these first sentences are written that made writing a robust grammar a challenge. The final grammar used in this algorithm preforms well for many articles, but is not perfect (see section 5 for more in depth analysis).

The algorithm then attempts to categorize the extracted word. Here categories are not words, but Wordnet synsets. I chose each category's associated synset based on which sense of the word I thought best encompassed the category. In general terms, the algorithm attempts to determine the most semantically similar category to the extracted noun. In order to do this, however, an appropriate synset must be chosen for the extracted noun. Initially this was done via NLTK's implementation of the Lesk algorithm.

The Lesk algorithm attempts to disambiguate a target word based on the assumption that the words in the target word's neighborhood will share a common topic. The best synset for a target word, then, is the one whose dictionary definition is most similar to the dictionary definition of other words in the sentence. While this makes sense in theory, in practice it was wildly inaccurate. I found that a brute force method, enumerating the synsets for a given word and comparing them all to each category synset, and choosing the highest scoring one, worked much better. Here semantic similarity is judged using NLTK's Lin similarity, a method similar to Resnik similarity, but one I found to be a litte more accurate.

5 Evaluation

In order to evaluate whether the algorithm worked, I ran the algorithm on three randomly chosen Wikipedia article. The name of the of each link, along with the extracted noun, and subsequent category are written to a line of a csv, for easy evaluation. I am using so few articles because the accuracy of the algorithm needs to be checked by hand, which is an extremely time-consuming process. Additionally, while I am only evaluating the results for three articles, across the articles, there were over 230 hyperlinks, which is a fair number of data points.

On average, I found that the algorithm chose the correct noun from the first sentence 59% percent of the time, but classified nouns correctly 49% percent of the time, for a given article. Note, however, that this average is slightly skewed, as one article had only 13 links, and another had over 170. If we look just at the total number of links classified, then correct noun extraction goes up to 64% percent, and correct noun classification to 51%. While the number of correct classifications seems a little low, one must take into account the fact that there are 13 possible classifications, which means 51% is about 6 times greater than chance. Indeed, Toral and Munoz (2006) wrote a similar NER algo-

rithm for Wikipedia and got as high as a 78% accuracy rate for categorization. However, Their algorithm was tailored to only two categories (PERSON and LOCATION; categories I found my algorithm to be best at given the lack of ambiguity and simplistic structure of most initial sentences associated with these type of articles).

Nevertheless, the algorithm appears to be much worse at classification then extracting the best noun. This is likely due to the way in which the algorithm determines the sense of the extracted noun. Word sense disambiguation is and incredibly difficult problem, and my algorithm attempts to do it in a 'braindead' manner. Humanlike word sense disambiguation is only possible by taking the word's context into account. I attempted this with the Lesk algorithm, but gave me worse results than the brute force approach.

6 Discussion

Coming into this project, I did not at all expect the noun extraction aspect to be so difficult. There are only so many nouns in a given a sentence, yet choosing which one best sums up an article is incredibly challenging. I ended up resorting to a lot of heuristical methods, which, by definition, only work some of the time. I was also not at all expecting this project to be so subjective. I had to make many seemingly arbitrary decisions, e.g., which category synsets to use, whether the algorithm categorized an article correctly, etc. It turns out NER is not as cut-and-dry a task as it initially appears, but actually is pretty intricate and abstruse. I also got to experience first hand the difficulty and monotony often part-and-parcel with NLP research. Combing through my results during the evaluation process was incredibly taxing and mind-numbing, and I cannot imagine what it must be like to have to hand annotate a corpus.

7 References

Toral, A., & Munoz, R. (2006, April). *A proposal to automatically build and maintain gazetteers for Named Entity Recognition by using Wikipedia.* In Proceedings of EACL (pp. 56-61).