**Introduction**

The problem this system solves is as follows: Given an entity from a ReVerb tuple, quickly come up with *k* Freebase entities that are likely candidates purely based on string matching. For example, for the tuple “(Clinton, visited, New York)”, top candidate entities such as Bill Clinton and Hillary Clinton should be returned. This system employs a variety of matching techniques such as exact string matching, exact cleaned string matching, full word substring matching, Wikipedia alias matching, acronym matching, and Lucene spell correction matching. On a test set of 93 unique terms from real news articles, our best system is able to return the correct match in the top 5 matches 83% of the time at a speed of 150 ReVerb entities per second. This is a 41% improvement over the naïve approach of using only exact string matches and ordering on prominence.

**Naïve Approaches**

A naïve approach to solving this problem would be using only exact string matches and ordering on matches; however, this misses many matches such as: “US” matching “United States”, “Obama” matching “Barack Obama”, and “Gadafy” matching “Muammar al-Gaddafi”. When we evaluated this approach, we found that it achieved 42% accuracy.

Another simple approach would be to just take substring matches, ordering on prominence. When we evaluated this approach coupled with exact string matching and prominence ordering, we found that it achieved 58% accuracy. Used alone, it provided only 20% accuracy. The reason it is so low is because pure substring matching provides many low quality matches that push the desired matches down past the 5 match threshold. Also this method still misses matches such as “US” matching “United States” and “Gadafy” matching “Muammar al-Gaddafi”.

These simple approaches are able to correctly match some types of matches, but they miss more complex types of matches. Table 1 enumerates a list of the different types of matching that a system might be required to handle in order to increase accuracy to a reasonable level.

Table 1: Outlines cases handled by our system.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | **Example Case** | | |
|  | **Match Type** | **ReVerb Entity** | **Freebase Entity** | **Handled?** |
| 1 | Exact match between arg1 and Freebase entity | Bill Clinton | Bill Clinton | Yes |
| 2 | Exact match between cleaned arg1 and cleaned Freebase entity | Titanic | Titanic (1997 film) | Yes |
| 3 | Arg1 is a substring of the Freebase match | Christ | Christian | Sometimes |
| 4 | Freebase match is a substring of arg1 | Christian | Christ | Sometimes |
| 5 | String overlap between arg1 and Freebase match | President Clinton | Bill Clinton | Yes |
| 6 | Arg1 is an abbreviation of a Freebase match | G.E. | General Electric | Yes |
| 7 | Misspelling match between arg1 and Freebase match | Gadafy | Muammar al-Gaddafi | Mostly |
| 8 | Prominent Alias between arg1 and Freebase match | America | United States | Mostly |
| 9 | Obscure Alias between arg1 and Freebase match | The Red Planet | Mars | No |

**Key Components in System**

In addition to the components listed below, we also incorporated a scoring algorithm for ordering matches. The naïve approaches ordered only on prominence, but this approach doesn’t take into account the higher quality of some types of matches over others. For example, an exact string match is higher quality than a single word-overlap match. All of the matching techniques listed below use case insensitive matching.

1. *Scoring Algorithm*:
2. *Exact String Matching*: Uses hash table stored in memory for fast lookup.
   1. Generally provides a few very high quality matches such as “Bill Clinton” matching “Bill Clinton”
3. *Exact Cleaned String Matching*: Freebase entities are cleaned, then loaded into a hash table for fast lookup of a cleaned ReVerb entity. Cleaning an entity involves removing commas, periods, quotes, html special characters, and removing parts of string in parentheses (“Titanic (1997 film)” would become “Titanic”)
   1. Handles cases like “Great Britain&rsquo” matching “Great Britain”.
4. *Wikipedia Alias Matching*: Uses a list generated from Wikipedia of different articles that refer to the same entity. For example, the article with the title “America” refers to the article with the title “United States”. Aliases are cleaned as in (2) and loaded in a hash table for fast lookup of a cleaned ReVerb entity.
   1. Handles cases like “America” matching “United States” or “Feb” matching “February”
5. *Word-Overlap Matching*: Determines how many words overlap between two strings. This method is better than pure substring matching for two reasons. One is that substring matching is much slower than this technique. For this method, a hash table can be loaded with each word for fast lookup of a cleaned ReVerb entity, while in substring matching; many string comparisons need to be made for each match. Another reason is that this method returns higher quality matches. Full word matches are generally better than partial word matches. Using only word-overlap matching achieves 57% accuracy, which is a 37% improvement over substring matching.
   1. Handles cases like “William Jefferson Clinton” matching “William Clinton”.
6. *Abbreviation Matching*: If a ReVerb entity looks like an acronym, then it is matched to the acronyms of all Freebase entities with at least 20 inlinks. A ReVerb entity looks like an acronym if it is composed of all capital letters. It could also have periods in between the letters, but is not required to. Acronyms for Freebase entities with more than 20 inlinks are computed prior to computation and loaded into a hash table for fast lookup. Freebase entities with fewer than 20 inlinks generally do not have prominent acronyms associated with them. On our dataset, all of the acronyms’ correct matches were found in the top 50,000 Freebase entities. The 20 inlink threshold includes around 300,000 of the top entities in Freebase.
   1. Handles cases like “U.S.” matching “United States” or “GE” matching “General Electric”
7. *Non-Plural Matching*: If a ReVerb entity ends with an “s”, it is stripped off and the stub is used to query against both the exact string hash table and the substring hash table. Plural entities are not handled by the Word-Overlap method, so a special case had to be made for them.
   1. Handles cases like “Egyptians” matching “Egyptian”.