Document Tag Clouds

Li Quan Khoo, Giulia Deiana

Intended audience:
BS/MEng/MSc CompSci

Introduction

Picture yourself browsing your favourite website with a fair bit of text. This can be news, blog posts, discussion threads, anything like that. These kinds of documents usually have a title, and sometimes the author or the site curator would manually tag it with additional topics contained within.

Our goal is to semi or fully automate topics or keyword identification in such documents. This is of particular relevance in discussion threads where the topic might drift away from the original title, or articles that change over time, such as a wiki article or any sort of collaborative writing. Using our method, one would simply re-index the documents to include the new topics or keywords to better represent them in search.

Abstract

We present a method that identifies the ‘tags’ in a document. There are three ‘families’ of tags.

1. Entity tags are names and proper nouns most often brought up in the article.
2. Keyword tags identify the most relevant topics in the document.
3. Lastly, domain tags tell the user what area (e.g. political, economic, scientific, etc.) the document belongs to.

$$f(D) = \{T_{ne}, T_{key}, T_{dom}\}$$

$$D1 \xrightarrow{f} \begin{cases} T_{ne} = \{\text{United States, Barack Obama, …}\} \\ T_{key} = \{\text{Election, candidate, poll, …}\} \\ T_{dom} = \{\text{Politics, …}\} \end{cases}$$

Figure 1. Our desired mappings

The keyword tags are language-agnostic, but the other tags generated by named entity recognition (NER) and from WordNet and domain information. NER performance and application depends on the language and training corpus, and WordNet dictionaries are in one language. Hence T_{key} and T_{dom} are inevitably language-specific.

For testing and evaluation, we used the Wiki10+ and the Reuters-21578 corpuses. Both are in English and are relatively clean, with very few noisy or misspelled text. We make use of WordNet and WordNet Domains ontologies, and the main NLP toolkit we used is NLTK.

Methodology

Assumptions

1. Our method assumes that misspelled words are few enough to be considered as background noise.
2. We assume that the classifiers used to perform the various NLP tasks such as NER and SBD are trained on a corpus with a similar language model to our corpus. Because this project is very short term, training them ourselves is outside its scope.

For a given document D, we want to label it with three sets of tags; named entity tags T_{ne}, keyword tags T_{key}, and domain tags T_{dom}.

To do this, we make use of two ontologies, WordNet and WordNet Domains. WordNet maps a string (which is not necessarily a monogram) into a list of WordNet senses, ranked by how often that particular sense is used.

For each WordNet sense, WordNet Domains gives at least one mapping to a domain label. WordNet domains are arranged in the form of a hierarchical tree graph of depth 4.

Entity tags with NER

Generating the set T_{ne} is simple. First, we perform sentence boundary disambiguation (SBD) on the corpus to get a list of sentences. We then identify the named entities (NE’s) in each sentence by using NLTK’s pre-trained NER classifier and rank them by their term frequency.

Keyword tags with RAKE

To generate candidate T_{key} tags, we use Rose et. al’s RAKE[1] algorithm. We used RAKE as it has been evaluated to be able to extract highly specific terminology. RAKE first segments the sentence into a list of candidate keywords, where each segment would be delimited by either a stopword, a phrase delimiter (most punctuation marks), or the sentence start or endpoint.

> Criteria of compatibility of a system of linear Diophantine equations, strict inequations, and nonstrict inequations are considered.

Figure 2. Bolded words are candidate keywords. Example from RAKE paper.

It then recovers potential keywords containing stopwords by looking for keywords that adjoin one another at least twice in the document, in the same order, but these are relatively rare. This has the advantage of being able to identify long keywords which might be missed by n-gram-based extraction.

The RAKE paper evaluates several metrics to identify which keywords are most important. Here, for simplicity, we rank them by term frequency as that does not involve constructing a word co-occurance matrix.

Domain tags with WordNet

For the domain tags, we make use of the WordNet and WordNet Domains ontologies.

To identify the likeliest word sense for each word in our list of keywords, we use the simplified Lesk Algorithm.[2] The algorithm takes adjacent words, and for each of their possible word senses, it calculates the amount of overlap between the word glosses, given a dictionary.

Once we have labeled each keyword with the likeliest word sense, we map it to its domain tag(s) in the WordNet Domains ontology, as well as all its ancestor nodes in the hierarchy.

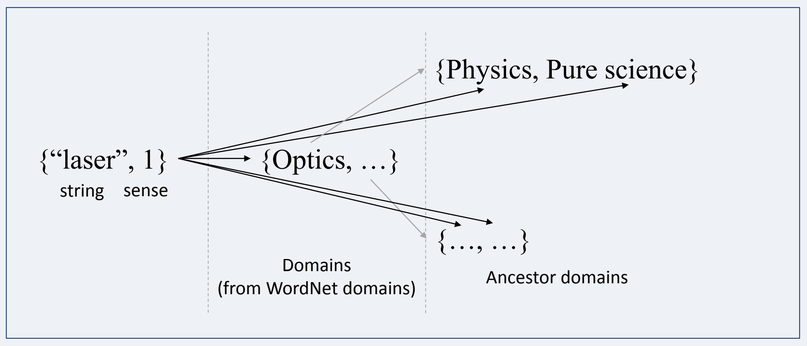

Figure 3. Word sense to domain mappings

Based on these mappings, we get a list of triples of the form {domain, count, depth}, and we can rank the most relevant domain labels based on a function of this triple, and selectively return or display the first n.

Finally, once we have identified all three kinds of tags, it becomes a simple matter of either labeling the documents with them, or reverse-indexing the documents with the tags, depending on what one wants to do.

Further work

It is outside the scope of this project, but we could attempt to map the identified keywords to their appropriate entities using an entity knowledge base such as YAGO or Wikidata, so we tag documents with the identified entity rather than just the string.

Bibliography

1. Rose et. al. (2010). Automatic keyword extraction from individual documents. Text Mining: Applications and Theory.
2. Lesk, M. (1986). Automatic sense disambiguation using machine readable dictionaries. ACM.
3. R. Navigli (2009). Word Sense Disambiguation: A Survey. ACM.