Content Extraction (Dina Hafez)

Introduction

In this problem, I have a Web page, and the goal is to extract the content of this page using Machine learning and NLP techniques.

The Code

The code was implemented in python 2.7

To run the code, from the command line write:

python KeyWordExtractor_supervised.py page_url [AlgorithmID] [noOfKeywords]

The following are optional parameters with default values

AlgorithmID: 1 = RAKE (default if no value is entered)
2 = Supervised Learning Model
3 = All two models

noOfKeywords : Number of keywords you would like to extract., defaults to 10 if no value is entered

Methodology

This is a keyword extraction problem in which the task is to automatically identify the term(s) that best describe the page. To do this, there are three main steps.

1. **HTML cleaning**: Since the Web page contains lots of HTML tags, a proper HTML parser should be used. Moreover, each Web page has lots of other div tags that are not related (like menu, sidebar, etc.). And since the main topic/keyword of the page is in the main text of the document, I used the readability package from python to return back the main content of each page. This package mainly returns the main body and the title of the page. I also used NLTK python package to clean the HTML tags and white spaces.

   Some information/keywords might be available in the page's metadata, but for the sake of this project, and time constraint, I ignored this input source.

2. **Candidate Identification**: In this step we take the cleaned full text and identify the candidate keywords, thus providing cleaner input for the machine learning model

3. **Key Phrase Extraction**: This is the machine learning model itself, in which we process the candidates keyword to get the final list of "real" keywords. Exploring the literature, I found multiple ways to handle this problem. Here, I tried three techniques and ended up using two of them.

I used two different models, one unsupervised model and the other is supervised model. I've defaulted the output to the unsupervised model yet the supervised model can be access in the code through an optional argument.

Note that there was another third model that builds graphs of connected words, using TextRank, based on this research [3]. Although the model seemed promising, the output was meaning-less and I wasn't sure if that's the problem with the model implementation or the model itself. So given the time constraint, I decided to ignore it, and focus on the other two models.

The following section will be a deep dive in the steps by which I handled this problem.

Candidate Identification

In this step, our objective is to improve the input to the machine learning model to provide better outcome instead of sending full text. Through some common sense rules and best practices, we can filter for candidate keywords, while safely excluding unnecessary words.

The following are the rules I applied on the text, in order to obtain a list of candidates.

1. The candidate phrase should not be considered as a stop word or a pronoun. For this, I filtered against the most commonly known stop words and pronouns using publicly available lists
2. The list of stop words is used to split the text into phrases, along with a sentence tokenizer. This means that all words in a candidate phrase should be contained in one sentence, and should not be separated by stop words.
3. Since a topic is expected to be a noun, the candidate phrase is selected to either be a one word that is noun, or up to 3-grams, but with at least one noun word.
4. if the candidate phrase contains more than 3 words, it is split into a sliding window of 3 words. For example a phrase like "2-Slice Compact Plastic Toaster" is split into two phrases "2-Slice Compact Plastic" and "Compact Plastic Toaster".
5. Finally, I check if all words in the phrase are numbers. If all of them are, the phrase is discarded.

Key-phrase Selection

This is the machine learning model(s) itself, in this section I've explored and implemented two models, the first is an unsupervised model, which provides better results now given the limited training, the second is a supervised model, which is can provide better results later should proper training and keyword corpus is used.

1. RAKE Algorithm

The first algorithm I explored is Rapid automatic keyword extraction (RAKE) algorithm [1]. This algorithm gives a score to a candidate based on the ration of degree of the candidate to the frequency of appearance. As explained in [1], degree of the word favors words that occur often and in longer sequence, while

words that occur frequently regardless of the number of words with which they co-occur is favored by the frequency metric. The ratio of degree to frequency favors words that predominantly occur in longer candidate keywords.

The output of this algorithm for the three test cases is shown in the result section.

2. Supervised Machine Learning Model

Another algorithm is a supervised learning model. The challenge here is how to get training examples. I was able to find a corpus Crowed500 [2]. This corpus includes text documents and annotated candidate keywords for each. A major issue is that these documents are not an exact representation to web pages, which definitely poses a big challenge, making the model not as accurate as one would want.

Coming up with the features for this model is a key parameter in accuracy of the output. Given the web pages provided as sample, I've decided to generate these features for each candidate keyword, There are two types of features, frequency based feature, which is a function of how often and when does the keyword appear, and structural features, which depends on the structure of the word.

Frequency-based features:

- Word frequency (normalized)
- Length of the candidate phrase
- Length of the longest word in the candidate phrase
- Lexical cohesion for candidate phrases with more than one word

Structural Features:

- Count of the number of words in the candidate phrase appearing in the title (This is a very important feature)
- Distance to the first occurrence of the candidate phrase (normalized)
- Distance to the last occurrence of the candidate phrase (normalized)
- Difference between the first and last occurrence

To train this supervised model, you need positive and negative classes (examples). Since the training corpus has the list of candidates, I randomly selected the negative set from the text document, I then extracted negative keywords to represent the negative class, with sample size equivalent to that of the positive candidate class. Negative sets were filtered to not include any of the positive examples and not a stop word. I then applied the feature extraction for the negative set.

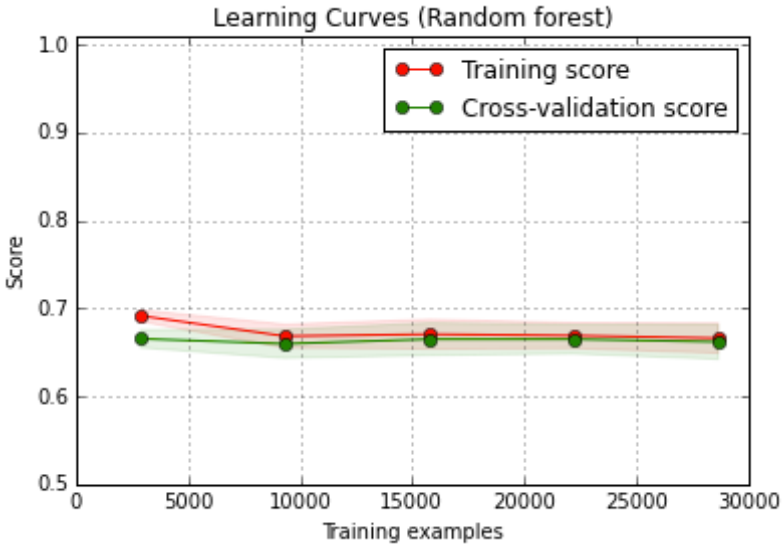
I trained a random forest classifier on this set. With 5 fold cross validation, this is the classification report resulted.

Classification Report:

	precision	recall	f1-score	support
-1.0	0.60	0.77	0.68	3888
1.0	0.78	0.61	0.68	5071

avg / total 0.70 0.68 0.68 8959

Looking at training curves, the model is experiencing high bias; implying that more features might help. Due to time constraints, I was not able to test more features.



Possible improvements

1. It's obvious that there is a big potential for supervised learning methods. However obtaining training data that matches the test data can be the next step in refining the process
2. Coming up with more relevant features that could be helpful that are relevant to a specific class of web pages, after test them within a feature extraction framework, would help identify which features are important to such problem.
3. Using features from external data sources, such as Wikipedia and calculating a frequency features based on how frequent the feature is could also be helpful.
4. For this problem, I built a binary classifier. Some literature is pointing at the importance of looking at the problem as a ranking problem. We would need to train a function to apply pairwise ranking for candidates.
5. Word expansion could also be of great added value. If we are to know which words in general are connected to candidate phrases, we could pick candidates with more connected terms or topics.

Results

	RAKE	**SupervisedModel**
Amazon	• compact plastic toaster • cpt122 2slice compact	• compact plastic toaster • 29 30 qty

	• conair cuisinart cpt122 • cuisinart cpt122 2slice • 2slice compact plastic • 29 30 qty • qty	
CNN	• president barack obama • secrets wide open • overactive mother teresa • cell phone calls • track cell phone • global speculation sunday • mother teresa gene • high school diploma • ultimate destination unknown • barack obama insists	• united states • united kingdom • undercover assets • ultimate destination unknown • track cell phone • told cnn affiliate • time snowden • terrorist attacks • spilling details • snowden told
Blog	• joyfully follow suit • bring tasty snacks • relationships require compromises • 20mile bike ride • steep fourhour hike • suggest overnight camping • short day hikes • short hike • day hike • day faster	• indoorsy friend • tricky terrain • summer night • strava record • short hike • short day hikes • shooting pool • novice buddy • live jazz • joyfully follow suit

References

1. Rose, Stuart, et al. "Automatic keyword extraction from individual documents." *Text Mining* (2010): 1-20.

2. https://github.com/snkim/AutomaticKeyphraseExtraction

3. Mihalcea, Rada, and Paul Tarau. "TextRank: Bringing order into texts." Association for Computational Linguistics, 2004.