Recommendation System For Bookmarks

The working of this system will be such that the bookmarks of a user are analyzed to find prominent keywords on that page. These keywords are then sent to a recommender system, which matches the keywords against a database of webpages. Webpages appropriate to the keyword is displayed to the user.

PART I : Keyword extraction from a user's bookmarks

We examine a wide variety of information sources such as title information, elements of headind and anchor tag, also the words of the url occuring on the page, in addition to the raw text body of the page.

- The entire html page of the particular bookmark is retrieved from the web.
- This html page is then passed through the BeautifulSoup parser to extract the elements of
  - the title tag
  - the heading tag
  - the anchor tag

- These words have higher probablity of being the keywords than the remaining raw data of the web page
- All these words are processed and eliminated from the raw text , in order to avoid redundancy and improve efficiancy of the system , taking into consideration the space and time complexity of the system
- A general methodology of text processing is followed for all the text included in the various tags mentioned earlier as well as the plain text of the webpage

Text processing

- The text is tokenized using the nltk word tokenizer

- Elimination of punctuation marks such as -.?,":;()`~!@#$%^*()_=+{}

- Removal of all the MS smart quotes to avoid problems of encoding

- This tokenized text is then passed to a part-of-speech tagger and a chunker
- lowercasing all the words
- forming a set out of these words
- elimination of stop words
- getting the term frequency

POS tagger :

The POS tagger used over here in based on classification. It gives an accuracy of about 93%. Features are extracted from words, then passed to an internal classifier. The classifier classifies the features and returns a label; in this case, a part-of-speech tag.

ClassifierBasedPOSTagger is a subclass of ClassifierBasedTagger that implements a feature detector that combines many of the techniques of previous taggers into a single feature set. The feature detector finds multiple length suffixes, does some regular expression matching, and looks at the unigram, bigram, and trigram history to produce a fairly complete set of features for each word. The feature sets it produces are used to train the internal classifier, and are used for classifying words into part-of-speech tags.

Basic usage of the ClassifierBasedPOSTagger is much like any other SequentialBackoffTaggger. You pass in training sentences, it trains an internal c classifier, and you get a very accurate tagger.

```
from nltk.tag.sequential import ClassifierBasedPOSTagger
tagger = ClassifierBasedPOSTagger(train=train_sents)
tagger.evaluate(test_sents)
```

output : 0.93097345132743359

Working:
ClassifierBasedPOSTagger inherits from ClassifierBasedTagger and only implements a feature_detector() method. All the training and tagging is done in ClassifierBasedTagger. It defaults to training a NaiveBayesClassifier with the given training data. Once this classifier is trained, it is used to classify word features produced by the feature_detector() method.

Cutoff probability:

Because a classifier will always return the best result it can, passing in a backoff tagger is useless unless you also pass in a cutoff_prob to specify the probability threshold for classification. Then, if the probability of the chosen tag is less than cutoff_prob, the backoff tagger will be used. Here's an example using the DefaultTagger as the backoff, and setting cutoff_prob to 0.3:

```
default = DefaultTagger('NN')
tagger = ClassifierBasedPOSTagger(train=train_sents,
backoff=default, cutoff_prob=0.3)
tagger.evaluate(test_sents)
```

output : 0.93110295704726964

So we get a slight increase in accuracy if the ClassifierBasedPOSTagger uses the DefaultTagger whenever its tag probability is less than 30%.

- Since there is a tendency for keywords to constitue noun phrases, we will put higher weightage on the terms that comprise noun phrases.
- Also since named entities such as person or organization names may be useful keywords , these are also extracted from the webpage. Two different NE systems preferably are used in order to provide coverage of a larger set of entity types
- Even among the noun phrases , the words that are capitalized have higher probablity of constituting the keyword
- The candidate set is filtered against a stoplist of most frequent English words
- Unigrams or bigrams containing a stopword are removed
- Candidate keyword extraction is used as an input for a classiefier which estimates the likelihood that a given phrase is a keyword

Classifiers:

- While training a classifier , better accuracy of candidate extraction helps to filter word combinations that are not likely keywords(such as stop words and non-noun phrases) and thus reduces the amount of negative training samples, thereby improving the ratio of positive to negetive training data

Note : The keyword extraction tast has an imbalance between positive and negetive samples , with very few positive label data

- **Keyword Classification unit :**
  - A classifier uses features of the candidate phrase to estimate the probablity that the phrase is a keyword , and assigns an output label (keyword or non-keyword) to the phrase. Along with this it ranks the keywords
  - The classifier function is obtained by using supervised machine learning
    Supervised machine learning : The mapping is learned by the classifier system based on a dataset where “correct” output labels have been provided by human annotators.
  - Maximum entropy classifier:
    - Derives constraints from the training data and assumes a distribution of maximum entropy in cases not covered by the training data
    - It consists of vectors of values (features such as term frequncy, inverse document frequency , etc.)for each keyword candidate , which are used by the model to learn the weights associated with each feature.
    - Given a new input data, the trained classifier can then compute probablity that a phrase is a keyword given the input values for that candidate phrase.

$$P(Y = 1|X = x) = \frac{\exp(\Sigma\alpha \cdot fi(x, c))}{1 + \exp\Sigma(\alpha \cdot fi(x, c))}$$

f = joint feature
α = weight assigned to that feature