Sentiment API Technical Documentation

Name	Size	Creator	Creation Date	Comment
File Lexicon_combined.csv	4.53 MB	laura.tolosi	Dec 08, 2014 16:45	Lexicon from three sources

Introduction

The aim is to evaluate sentiment polarity (Negative/Positive) at several levels of granularity:

- **document** (overall sentiment): appropriate for blog posts or technical review articles, estimates whether the author's opinion on the topic is generally positive or negative. Strong polarity means the author is very subjective.
- **paragraph** (aspect oriented): each paragraph expresses an aspect of the overall topic discussed in the document. News articles that are bound to present a balanced view on an event, are expected to alternate positive and negative aspects in the constituent paragraphs.
- **entity** (very specific target): appropriate for extracting detailed opinion on products and components, events, and others, together with aggregation over a corpus, for market analysis.

Sentiment prediction can be supervised, semi-supervised or unsupervised.

Supervised approaches rely on annotated datasets. Given the strong domain specificity, it is important that a large corpus from the target domain is available. When not available, domain adaptation methods can be used, that rely on a large out-of-domain corpus and a small supplementary target-domain annotated corpus.

Unsupervised methods rely on sentiment dictionaries: large lists of words with scores quantifying their polarity. Mapping to dictionary and aggregation statistics are used to evaluate sentiment in free text.

Semi-supervised approaches rely on a small set of annotated texts or small polarity dictionaries, that are expanded by either bootstrap methods, or by using external knowledge-bases like Wordnet.

Our approach (for English)

Our tagging services in intended for generic documents, without specified domain (at least in the early stages). Therefore we opted for an unsupervised approach. We composed a large sentiment dictionary from several open sources, as described below.

Sentiment dictionary

We assembled a sentiment dictionary from three sources:

1. SentiWordNet: http://tcc.itc.it/projects/ontotext/sentiwn.html (small)
2. MPQA opinion corpus: http://www.cs.pitt.edu/mpqa/ (large)
3. Stanford IMDB review dataset: http://ai.stanford.edu/~amaas/data/sentiment/ (very large)

From each of the above sources we extracted scores in an unique format, namely one score that is between 0 and 1, where the polarity is positive if score is close to 1, and negative, if it is close to 0. It can be expressed also as two scores, positive and negative, that sum up to 1.

SentiWordNet processing:

Terms of SentiWordNet are assigned polarity scores in dependence of their synset. Therefore, one term can occur several times, with different meanings and different polarity scores. We aggregated the scores into one, as follows:

- words with both high positive and high negative scores (in different synsets) clearly depend on the context; they were not many so we eliminated them, because otherwise sense disambiguation would have been necessary. These words were defined as follows:
  min_synsets(sentiwordnet_Pos(w)-sentiwordnet_Neg(w)) < -0.5
  max_synsets(sentiwordnet_Pos(w)-sentiwordnet_Neg(w)) > 0.5

- words with very similar positive and negative score in all synsets are said to be neutral. We also remove them: max_synsets(abs(sentiwordnet_Pos(w)-sentiwordnet_Neg(w))) < 0.2
- the final score is the most polarizing difference in a synset and map it to [0,1]
  score_sentiwordnet(w) = 0.5 max_synsets(sentiwordnet_Pos(w)-sentiwordnet_Neg(w))+0.5

MPQA processing:

The dataset is annotated with positive, negative and neutral, without probabilities. We assigned:
w positive, **score_MPQA(w) = 1**
w negative, **score_MPQA(w) = 0**
w neutral, **score_MPQA(w) = 0.5**

IMDB processing:

We obtain probabilities from counts as follows:

score_IMDB = P(positive|w) = count(w in positive documents) / count(w in documents)

Aggregation into one final score:

score(w) = 0.4 score_SentiWordNet(w) + 0.4 score_MPQA(w) + 0.2 score_IMDB(w)

The resulting file is attached to this page.

Sentiment evaluation algorithms

Pipeline for document sentiment:

1. Tokenization (+ stemming)
2. Mapping to dictionary
3. Sentiment evaluation: average over the scores of all mapped words

Pipeline for paragraph:

1. Tokenization (+ stemming)
2. Mapping to dictionary
3. Paragraph identification
4. Averaging scores of mapped words per paragraph

Pipeline for entity sentiment:

1. Concept tagging
2. Tokenization
3. Segmentation: using parsing, identify which tokens refer directly to the target entity (not available in the current version)
4. Map tokens to the senti-dictionary
5. Sentiment evaluation for the target entity:

- If segmentation is performed, average over the scores of tokens that are related to the target entity
- Otherwise, use an aggregate score that gives larger weight to the close-by tokens, rather than the remote ones (in the frame of a sentence):
  score (E) = sum_(w in sentence) score(w)/dist(w,E)