Analyzing Similarity

Exploring two similarity techniques, TF-IDF and Jaccard Distance

Outline

- TF-IDF
  - Term Frequency – Inverse Document Frequency
- Jaccard Distance

Frequency Analysis

- Last Analysis Interlude
  - Generated term frequencies
  - Tracked (charted) term frequency change over a week
- How different is enough difference?

Frequency Analysis

- Last Analysis Interlude
  - Generated term frequencies
  - Tracked (charted) term frequency change over a week
- How different is enough difference?
- Frequency alone is not enough
  - Normalization can help
    - Normalize over what

Collections/Corpus

- Collection or a corpus
  - This is some set of documents
- Can we use terms to identify one or more documents in the collection?
- The frequency of the terms in a document could be used to find the document

TF-IDF

- Term Frequency – Inverse Document Frequency
  - We normalize terms in each document
  - The rarity/commonality of term helps distinguish one document from another

Tweet Corpus/Documents

- What is the Document?
  - A single tweet?
  - An hour of tweets?
  - A day of tweets?
  - All tweets by a single person?
- What is the Corpus?
  - The complete set of tweets

Sample Code

- Meeting Schedule page
  - explore_tfidf.py

- Bits of code that help build tf-idf
- This sample code is usable, but NLTK (Natural Language Tool Kit) has implementations of this too

Demo code

- Procedures
  - make_doc() – doc data structure
  - build_corpus() – corpus data structure
  - tf() – calculate normalized term frequency
  - idf() – calculate inverse document frequency
  - tf_idf() – calculate the tf-idf based on tf() and idf()
  - doc_top_n() – list top N terms of the doc
  - doc_has() – check whether a doc has a given term

Demo

- explore_tfidf.py

Using TF-IDF

- TF-IDF is really for searching and finding documents in a corpus
- TF-IDF can be thought of as a similarity measure
  - Given a set of terms, which documents in the corpus are most similar?
  - A cluster of related documents
- The current code does not do that, probably useful to explore that issue

Other Similarity Measures

- TF-IDF is a limited similarity measure
- Cosine similarity
  - Vector space model
    - Do two vectors point in the same direction?
  - Code in explore_tfidf.py could be used to create scored term vectors (described in the book, Russell)
- Jaccard Similarity (Jaccard Distance)

Jaccard

- How similar are these strings?

```
text1 = "this is a string of text that has words in it"
text2 = "this string also has some words, but it is different"
text3 = "other text might have stuff, if strings were what we test"
```

Jaccard

- How similar are these strings?

```
text1 = "this is a string of text that has words in it"
text2 = "this string also has some words, but it is different"
text3 = "other text might have stuff, if strings were what we test"
```

- Maybe use the number of tokens (words) that are the same?

Jaccard

- How similar are these strings?

```
text1 = "this is a string of text that has words in it"
text2 = "this string also has some words, but it is different"
text3 = "other text might have stuff, if strings were what we test"
```

- Maybe use the number of tokens (words) that are the same?

- 1.0 – (#_of_the_same_tokens / total_#_of_unique_tokens)

Quick little test of our intuition

```
from nltk.metrics.distance import jaccard_distance

text1 = "this is a string of text that has words in it"
text2 = "this string also has some words, but it is different"
text3 = "other text might have stuff, if strings were what we test"

print jaccard_distance(set(text1.split()),set(text2.split()))
print jaccard_distance(set(text2.split()),set(text3.split()))
print jaccard_distance(set(text1.split()),set(text3.split()))
```

Demo Code

- Procedures
  - query_date() – same as before, query the DB
  - create_tid_dict() – create a dictionary of tweet id and tweet
  - get_comparison_text() – return text to use for distance comparison
  - clustered_key() – return whether or not this key has been clustered
  - dump_cluster_info() – print out some information about the clusters
  - build_tweet_cluster() – actually build the clusters

Demo

- explore_jaccard.py

Quick Summary

- Explored
  - TF-IDF – get term based scores
    - Can think of this as a term based clustering. Given a set of terms, which documents are closest to those terms. Sometimes we call those terms a "query"
  - Jaccard Distance
    - What is the term based overlap between two documents? We can cluster based on this measure
    - Trouble with short text (aka tweets)

Other Similarity Metrics

- Cosine similarity
  - Mentioned this one earlier in the lecture
- Student's t-score (or Chi-Square test)
  - Used as an n-gram (bi-gram, tri-gram) measure – assumes a normal distribution of the co-occurance of words
- Edit Distance (Levenshtein distance)
  - How many one character edits are needed to change one string into another?
  - This might be good for short text, like tweets (given some data cleaning)