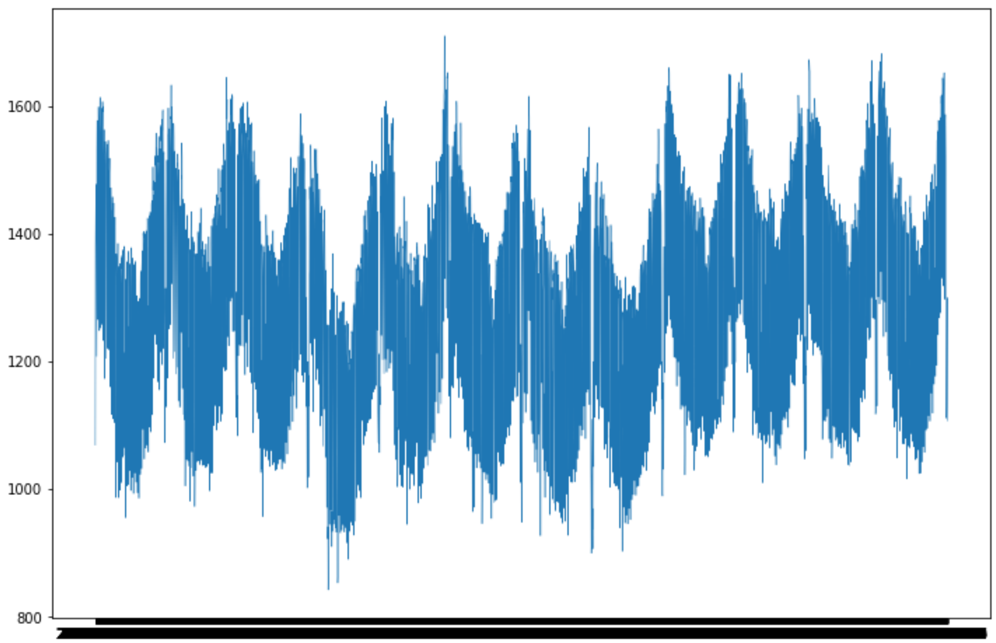
Lec05. Data Analytics for Texts

Recap: Time Series Analysis

- **Exploratory data analysis:** examine a time series with a line chart or statistics.
- **Segmentation:** Splitting a time-series into a sequence of segments. Represent a time-series as a sequence of individual segments, each with its own characteristic properties.
- **Classification:** Assigning time series pattern to a specific category → Natural phenomena, astronomical phenomena, animal movement.
- **Forecasting:** is the use of a model to predict future values based on previously observed values.
- **Decomposition:** deals with complex timeseries to help forecasting easier
- **Anomaly Detection:** Finding outlier data points relative to some standard or usual pattern.

Recap: EDA of Time Series

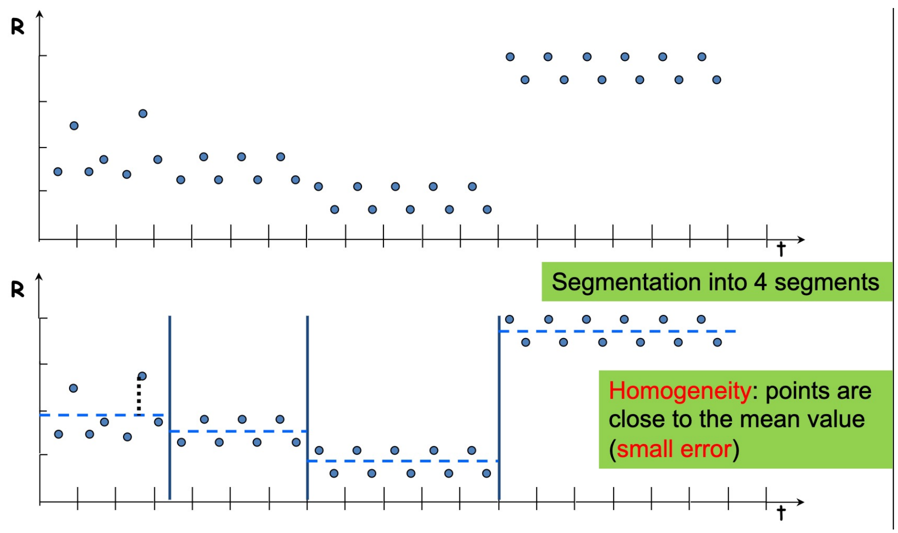
- Examine a time-series with statistics:
  - **Pros:** Summarize the values
  - **Cons:** do not consider the timestamps



count	4383.000000
mean	1338.675836
std	165.775710
min	842.395000
25%	1217.859000
50%	1367.123000
75%	1457.761000
max	1709.568000

Recap: Time Series Segmentation

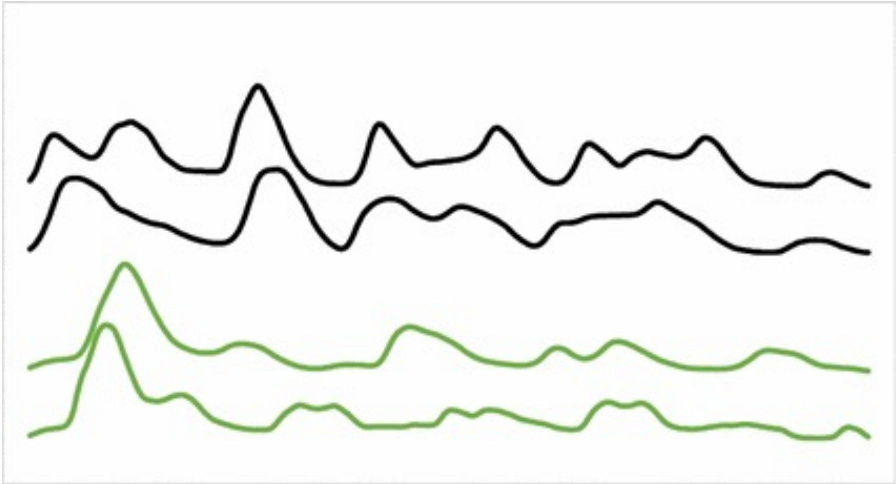
- **Goal:** discover structure in the time series and provide a concise summary
  - Useful for really long time series
  - Divide and conquer: Make it easier to analyze
- **How:** given a time series S, segment it into K disjoint segments (or partitions) that are as homogeneous as possible
  - Data points in the same segment are ``similar''
  - Similar to clustering but only allow grouping along the time dimension



Recap: Time Series Classification

- Assigning time series pattern to a specific category.
  - Given a time series $\mathrm{Y} = (y_1, y_2, \ldots, y_n)$ and a list of categories $C = (c_1, c_2, \ldots, c_k)$, we want to assign Y to it the best matching category c_i.
  - Needs a similarity/distance measure.

- **Applications**:
  - Identify a word based on series of hand movements in sign language.
  - Handwriting classification.
  - Moving pattern similarity.



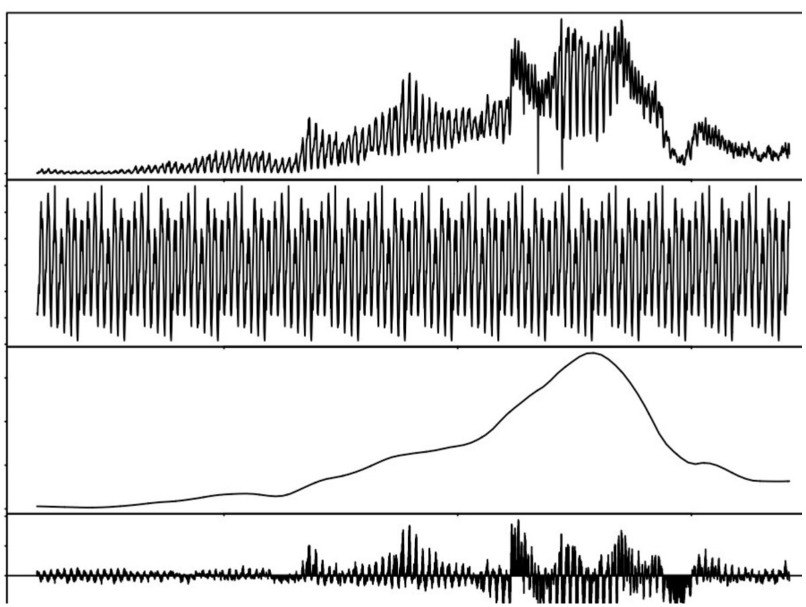
Recap: Forecasting with Seasonality

- Step 1. Calculate the average demand for each year
- Step 2. Calculate seasonal indexes
- Step 3. Average the indexes
- Step 4. Forecast demand for the next year
- Step 5. Multiple next year's average seasonal demand by each average seasonal index

Quarter	Year 1	Seasonal Index	Year 2	Seasonal Index	Avg. Index	Year3
Fall	24000	**1.2**	26000	**1.24**	**1.22**	**26840**
Winter	23000	**1.15**	22000	**1.05**	…	**…**
Spring	19000	**0.95**	19000	…		
Summer	14000	**0.7**	17000	…		
Average	**20000**		**21000**			**22000**

Recap: Time Series Decomposition

- Sometimes, a time series is too complex for segmentation, classification, or forecasting
  - It is better to understand short-term, long-term and recurring patterns first
  - **Approach**: decompose a time series into several components, each representing one of the underlying patterns.



Original time series =

Seasonality component +

Trend component +

Residue component

Recap: Time Series Anomaly Detection

- Finding **outlier** data points relative to some standard or usual pattern.
  - Such as unexpected spikes, drops, trend changes and level shifts.
  - Basically, an anomaly detection algorithm should either **label** each time point with *anomaly/not anomaly*, or **forecast** a signal for some point and test if this point value varies from the forecasted enough to deem it as an anomaly.

Examples:

- Growth of users in a short period of time that looks like a spike.
- When your server goes down and you see zero or a really low number of users for some short period of time.

Course structure

W1. Data Processing with Python

W2. Data Exploration with Python

W3. Data Modeling with Pytyhon

W4. Data Analytics for Timeseries

W5. Holiday

W6-7. Data Analytics for Texts

W8. Data Analytics for Images

W9. Data Analytics for Graphs

W10-11. Data Analytics for Other Data

W12. Revision

Textual Data Analytics: Application

Social Media Analytics

1. **Brand Reputation Monitoring**
   - Social Media, Blogs, News sites
2. **Advertising Performance Metrics**
   - Social Media, Blogs

CRM

1. **Complaint Tracking**
   - Social Media, Blogs, News
2. **Call Center Analytics**
   - Call Center Transcripts
3. **Competitive Analysis**
   - Communication, Surveys
4. **Market Research**
   - Surveys, Feedback

Predictive Analytics

1. **Prediction of Stocks**
   - Financial News, Newspapers
2. **Prediction of Election Results**
   - Social Media
3. **Movie Intake**
   - Twitter

Textual Data

- **Characteristics of textual data:**
  - Unstructured
  - Building blocks are words
    - Words are not independent
  - Each text segment (e.g. sentence) encapsulates semantics behind

Textual Data Analytics

- **Characteristics of textual data:**
  - Unstructured
  - Building blocks are words
    - Words are not independent
  - Each text segment (e.g. sentence) encapsulates semantics behind

Syntactical Analysis: transform unstructured text to structured representation

Semantic Analysis

Textual Data Analytics

I. Syntactical Analysis
   - ➢ Feature engineering
   - ➢ Representation learning

II. Semantic Analysis
   - ➢ Sentiment Analysis

I. Syntactical Analysis

- Transform a textual data into a multi-dimensional vector
- **Approaches:**
  - Feature Engineering: hand-craft the features
  - Representation learning: auto-learn the features (e.g. neural embedding)
- **Applications**
  - Information retrieval: search relevant documents given a query
  - Classification: categorize a text into a pre-defined label
    - e.g. spam email detection, Gmail tabbed categories
  - …

Syntactical Analysis Pipeline

Input text → Text Normalization → Feature extraction

Strings | Bag-of-words model | Vector representation

Example:

Document 1

"The goal is to turn data into information, and information into insight"
Carly Fiorina

Document 1

"The **goal** is to turn **data** into **information**, and **information** into **insight**"
Carly Fiorina

goal v_1
data v_2
information
…
insight v_W

Text Normalization

- **Remove stop words**
  - the, of, and, to, …(typically about 400 to 500 such words in English)
- **Stemming:** techniques used to find out the root/stem of a word:
  - e.g. use is the root of user, users, used, using
  - e.g. engineer is the root of engineering, engineered, engineer
- **Removing special characters and symbols**
  - E.g. !, .
- **Expanding contractions**
  - Contractions are shortened version of words or syllables
  - E.g., isn’t → is not, you’re → you are
  - Exist extensively and pose a problem to text analytics

Feature extraction: Feature engineering

Hand-craft the features

- **Term frequency:** Most common form of representation in feature engineering is the *term - document* matrix

Terms	Doc1	Doc2
goal	1	0
data	1	2
information	2	2
insight	1	0
you	0	1

Document 1

"The goal is to turn data into information, and information into insight"

Carly Fiorina

Document 2

"You can have data without information, but you cannot have information without data."

Daniel Keys Moran

$$doc_1 = \begin{bmatrix} 1 \\ 1 \\ 2 \\ 1 \\ 0 \end{bmatrix} \quad doc_2 = \begin{bmatrix} 0 \\ 2 \\ 2 \\ 0 \\ 1 \end{bmatrix}$$

document

term

$$A = \begin{array}{c l} \left[\begin{matrix} 1 & 0 & \cdots \\ 1 & 2 & \cdots \\ 2 & 2 & \cdots \\ 1 & 0 & \cdots \\ 0 & 1 & \cdots \end{matrix}\right] & \begin{matrix} \text{goal} \\ \text{data} \\ \text{information} \\ \\ \text{you} \end{matrix} \end{array}$$

term-document matrix

Term frequency: Another example

Example: 10 documents: 6 terms

	Database	SQL	Index	Regression	Likelihood	linear
D1	24	21	9	0	0	3
D2	32	10	5	0	3	0
D3	12	16	5	0	0	0
D4	6	7	2	0	0	0
D5	43	31	20	0	3	0
D6	2	0	0	18	7	6
D7	0	0	1	32	12	0
D8	3	0	0	22	4	4
D9	1	0	0	34	27	25
D10	6	0	0	17	4	23

$$D_1 = (d_{i1}, d_{i2}, \ldots, d_{it})$$

Each document now is just a vector of terms, sometimes boolean.

Term frequency: Normalize by tf-idf weights

- More information beyond word counts
  - Not all terms are of equal importance
  - If a term occurs frequently in many documents it has (more/less?) discriminatory power

Term frequency: Normalize by tf-idf weights

- More information beyond word counts
  - Not all terms are of equal importance
  - If a term occurs frequently in many documents it has less discriminatory power
- One way to correct for this is inverse-document frequency (IDF).
  - **tf**: term frequency - number of term occurrences in a document
  - **idf:** inverse document frequency - how much information the term provides in corpus C.
    - $idf(t, C) = \log \frac{|C|}{|C_t|}$, where
      - $|C|$: the number of documents in the corpus
      - $|C_t| = |\{d \in C : t \in d\}|$: the number of documents containing term t
    - More documents contain term t, less information it provides ($idf \rightarrow 0$)
  - **tf–idf**:
    - $tfidf(t, d, C) = tf(t, d) \times idf(t, C)$

TF-IDF example

Document 1

"The goal is to turn data into information, and information into insight"

Carly Fiorina

Document 2

"You can have data without information, but you cannot have information without data."

Daniel Keys Moran

term frequency (tf)

Terms	goal	data	information	insight	you
Doc1	1	1	2	1	0
Doc2	0	2	2	0	1

document frequency (df)

Terms	goal	data	information	insight	you
df	1	2	2	1	1

inverse document frequency (idf)

Terms	goal	data	information	insight	you
idf	0.69	0	0	0.69	0.69

$\log\frac{2}{1}$ $\quad$ $\log\frac{2}{2}$

tfidf

Terms	goal	data	information	insight	you
Doc1	0.69	0	0	0.69	0
Doc2	0	0	0	0	0.69

$$\frac{0.69}{\sqrt{0.69^2+0.69^2}}$$

tfidf (l2 normalized)

Terms	goal	data	information	insight	you
Doc1	0.71	0	0	0.71	0
Doc2	0	0	0	0	0.69

TF-IDF: Another Example

TF

	Database	SQL	Index	Regression	Likelihood	linear
D1	24	21	9	0	0	3
D2	32	10	5	0	3	0
D3	12	16	5	0	0	0
D4	6	7	2	0	0	0
D5	**43**	31	20	0	3	0
D6	2	0	0	18	7	6
D7	0	0	1	32	12	0
D8	3	0	0	22	4	4
D9	1	0	0	34	27	25
D10	6	0	0	17	4	23

TF IDF

	Database	SQL	Index	Regression	Likelihood	linear
D1	2.53	14.6	4.6	0	0	2.1
D2	3.3	6.7	2.6	0	1.0	0
D3	1.3	11.1	2.6	0	0	0
D4	0.7	4.9	1.0	0	0	0
D5	4.5	**21.5**	10.2	0	1.0	0
D6	0.2	0	0	12.5	2.5	11.1
D7	0	0	0.5	**22.2**	4.3	0
D8	0.3	0	0	15.2	1.4	1.4
D9	0.1	0	0	**23.56**	9.6	17.3
D10	0.6	0	0	11.8	1.4	16.0

Term frequency: Normalize by tf-idf weights

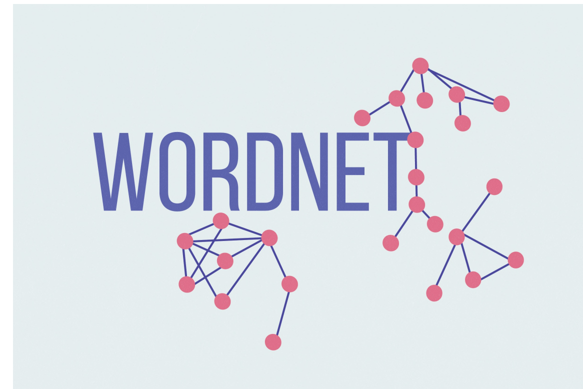
- Question 1: what happens if a term t appears in all documents?
- Question 2: what happens if term t is not in the corpus?

Term frequency: Normalize by tf-idf weights

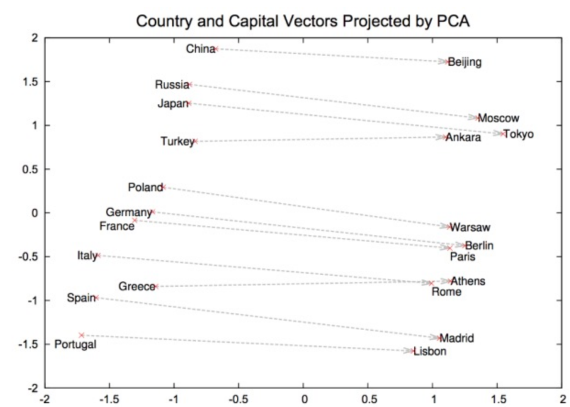
- Question 1: what happens if a term t appears in all documents?
  - idf = 0 → that term is just redundant like a stop-word
- Question 2: what happens if term t is not in the corpus?
  - i.e. $|C_t| = 0$ → cannot compute idf
  - One solution - smoothing
    - $idf(t, C) = \log \frac{1+|C|}{1+|C_t|}$

2. Representation Learning

- Words in a document are not independent, but stand in a semantic relation to one another.



- Word embedding: neural embedding and vector representation of words
  - Similar words will stay closer
  - State-of-the-art: word2vec



word2vec

[Mikolov et al. 2013]

Word embedding: Motivation

- Traditional encoding has no meaningful comparison rather than equality testing.
- Word2vec: capture some context via similarity

```python
import numpy as np
from numpy import dot
from numpy.linalg import norm

# compute cosine similarity for two vector u & v
def cosine_sim(u, v):
    return dot(u, v)/(norm(u)*norm(v))

vocabulary = ['king', 'man', 'queen', 'woman']
tokens = {w:i for i,w in enumerate(vocabulary)}

N = len(vocabulary)
W = np.zeros((N, N))
np.fill_diagonal(W, 1)


print ("cosine_similarity('king','woman'): {}".format(cosine_si
print ("cosine_similarity('man','woman'): {}".format(cosine_sim
print ("cosine_similarity('queen','woman'): {}".format(cosine_s
```

```
cosine_similarity('king','woman'): 0.0
cosine_similarity('man','woman'): 0.0
cosine_similarity('queen','woman'): 0.0
```

Without word2vec

```python
from gensim.models import Word2Vec

# load pre-trained GoogleNews model
model_file = 'model/GoogleNews_small'
W = Word2Vec.load(model_file)

print ("cosine_similarity('king','woman'): {}".format(cosine_sim(W['king'], W['woman'])))
print ("cosine_similarity('man','woman'): {}".format(cosine_sim(W['man'], W['woman'])))
print ("cosine_similarity('queen','woman'): {}".format(cosine_sim(W['queen'], W['woman'])))
```

```
cosine_similarity('king','woman'): 0.128479748964
cosine_similarity('man','woman'): 0.766401290894
cosine_similarity('queen','woman'): 0.316181391478
```

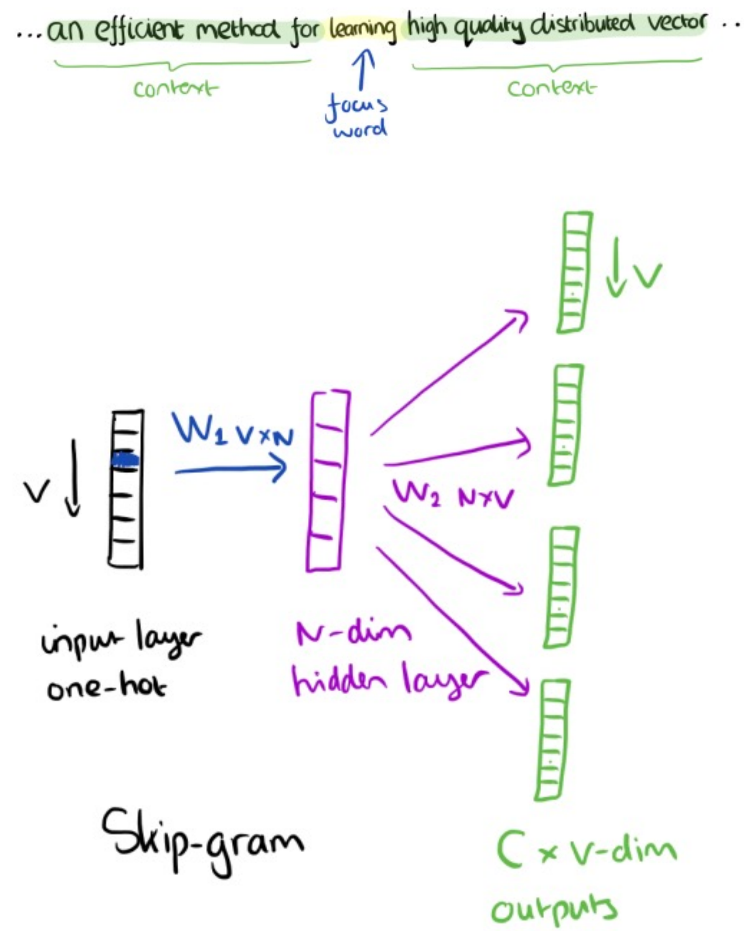
With word2vec

Word embedding: How it works

- **Word2vec** (Mikolov et al. 2013):
  - Learn a real-valued vector for each word.
  - Small distances induce similar words.
- Learning principle:
  - "You shall know a word by the company it keeps" - J.R. Firth 1957
  - Two models implement this principle:
    - Predict surrounding words given a centre word (Skip-gram model)
    - Predict a centre word given its surrounding context words (Continuous Bag of Words model - CBOW) .

Word embedding: How it works

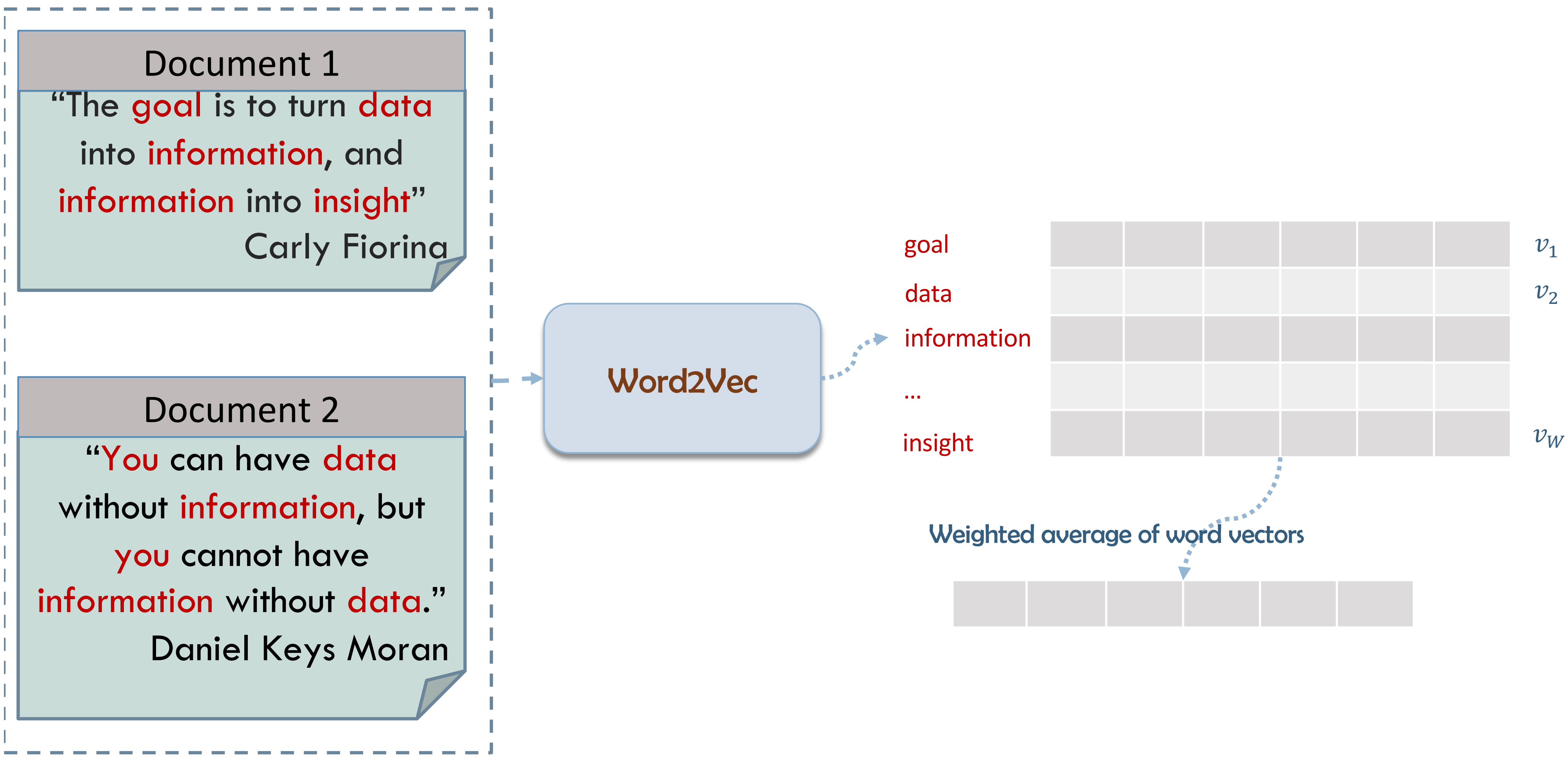
- Key steps:
  1. Take large corpus of text, represent each word as “one-hot” vector (dim=size of vocabulary)
  2. Build a neural network with input is the vector of focus word and output is a list of vectors of context words
  3. Train the neural network parameters (i.e. W_1, W_2) by passing all words in the corpus
  4. The final word vector for each word is its vector in the hidden layer after learning the parameters.



predicting the context given a word

Word embedding for documents

- Transform a document to vector by
  - Average its word vectors, weighted by word frequency in the document



Word2vec for documents

```python
CORPUS = ['the sky is blue','sky is blue and sky is beautiful',
'the beautiful sky is so blue','i love blue cheese']
# tokenize corpora
TOKENIZED_CORPUS = [nltk.word_tokenize(sentence) for sentence in CORPUS]
# build the word2vec model on our training corpus
model = gensim.models.Word2Vec(TOKENIZED_CORPUS, size=10, window=10,min_count=2, sample=1e-3)
```

Declare documents, train Word2vec model

```python
# define function to average word vectors for a text document
def average_word_vectors(words, model, vocabulary, num_features):
    feature_vector = np.zeros((num_features,),dtype="float64")
    nwords = 0.
    for word in words:
        if word in vocabulary:
            nwords = nwords + 1.
            feature_vector = np.add(feature_vector, model[word])
    if nwords:
        feature_vector = np.divide(feature_vector, nwords)
    return feature_vector
```

Computer average vector for a document

```python
def averaged_word_vectorizer(corpus, model, num_features):
    vocabulary = set(model.wv.index2word)
    features = [average_word_vectors(tokenized_sentence, model, vocabulary,num_features)
                for tokenized_sentence in corpus]
    return np.array(features)
```

Computer average vector for the entire corpus

```
[[ 0.011 -0.028 -0.002  0.008 -0.01   0.03  -0.002 -0.028 -0.001 -0.001]
 [ 0.001 -0.036  0.009  0.003 -0.007  0.036 -0.014 -0.011  0.001  0.009]
 [ 0.001 -0.024  0.005  0.013 -0.015  0.028 -0.008 -0.013  0.005  0.001]
 [ 0.02  -0.031 -0.027  0.011  0.009  0.014  0.044 -0.023 -0.018  0.048]]
```

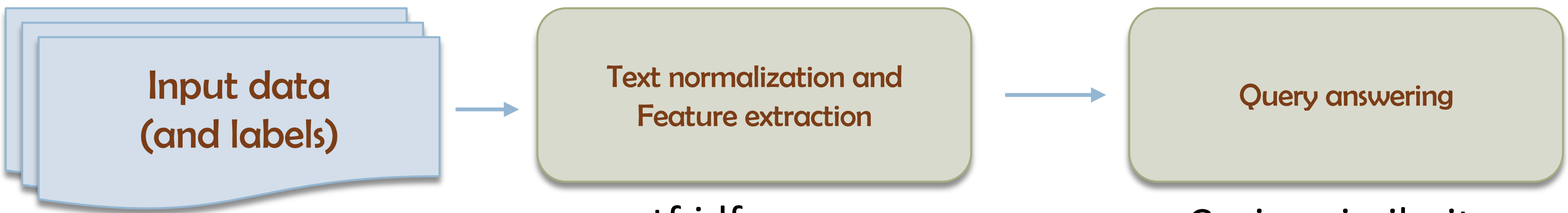
Output vectors (each document in each line)

3. Application: Information Retrieval

- There is a large corpus of text documents, and I want the one closest to a specified query
  - E.g. web search
  - A query is a representation of the user's information needs
    - Query can be a simple question in natural language
    - Normally a list of words.

- **Problem:** Find k documents in the corpus which are most similar to my query.
  - Queries can be represented as a vector in the same space
    - e.g. "Database Index" = (1,0,1,0,0,0) (if using tf-idf)
  - Solution: rank documents by the distance between vector representation of the query and the document vectors

Information Retrieval: Pipeline



- tf-idf
- word2vec

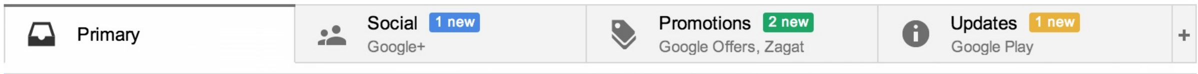
- Cosine similarity
- Euclidean similarity
- …

❖ Evaluation:

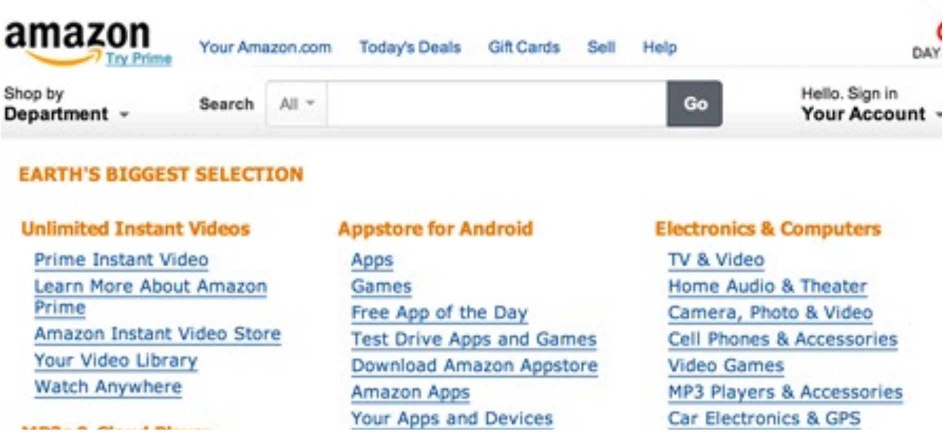
- Precision: the percentage of retrieved documents that are in fact relevant to the query (i.e., "correct" responses)
- Recall: the percentage of documents that are relevant to the query and were, in fact, retrieved
- …

Application: classification

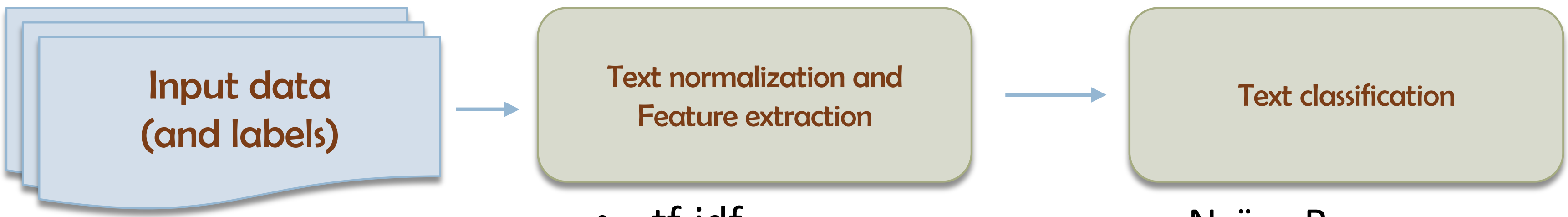
- Also called text categorization
- Given the following:
  - A training set of labeled text objects, and
  - A set of predefined set of classes (or categories)
- Applications
  - Spam email detection/filtering
  - Gmail tabbed categories



  - Tagging content or products using categories



Text Classification: pipeline



- tf-idf
- word2vec

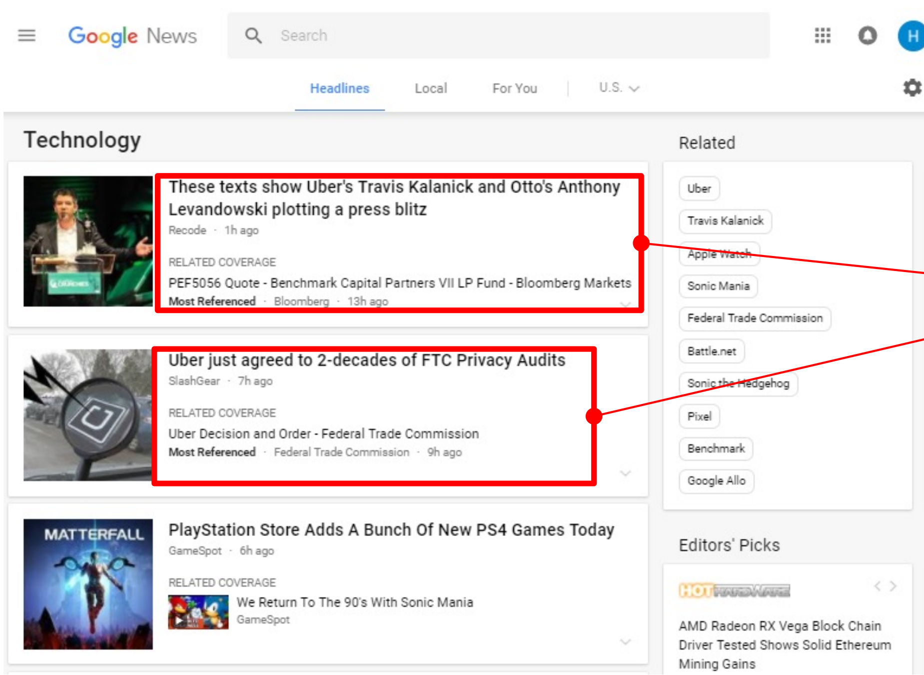
- Naïve Bayes
- Logistic Regression
- SVM
- Decision Tree
- Random Forest

❖ Evaluation:

- Confusion matrix
- Accuracy
- Precision
- Recall
- F1 measure

Application: Clustering

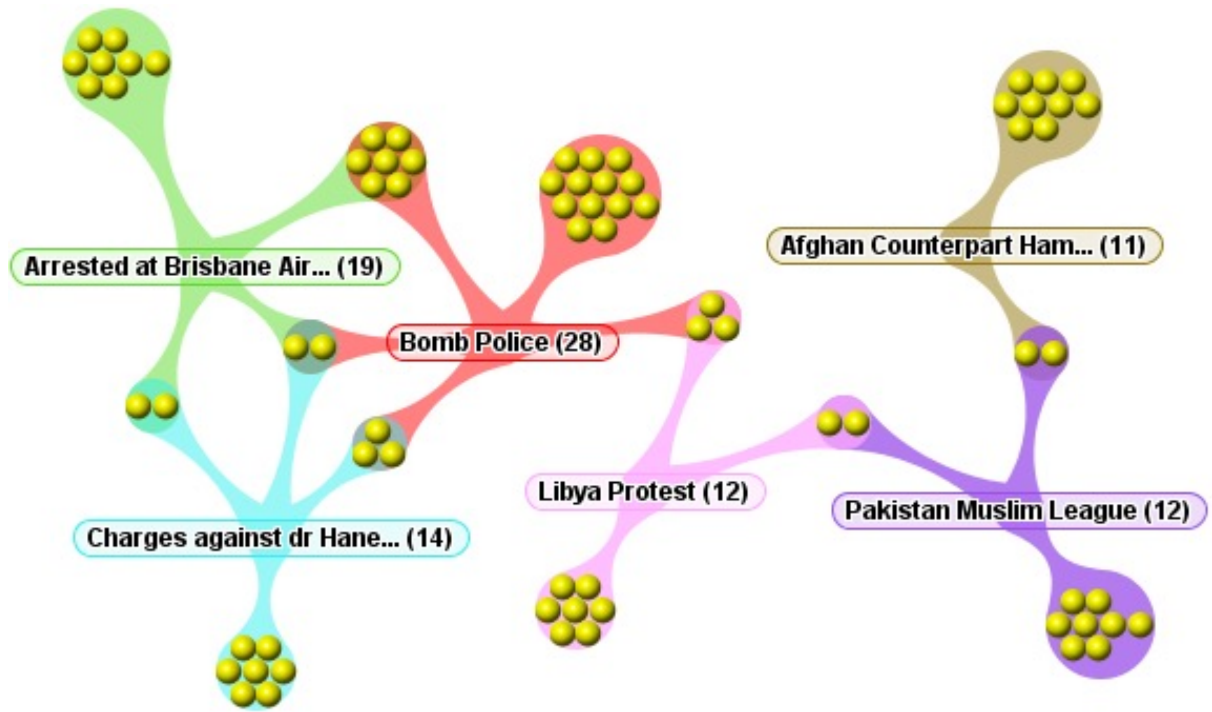
- What to do if no labels available for documents?
  - Can we still make sense out of these unlabelled text copora?
  - And discover "natural structure"
- Clustering: group similar texts together
- Applications:
  - Google News: automatic clustering related news



Related news

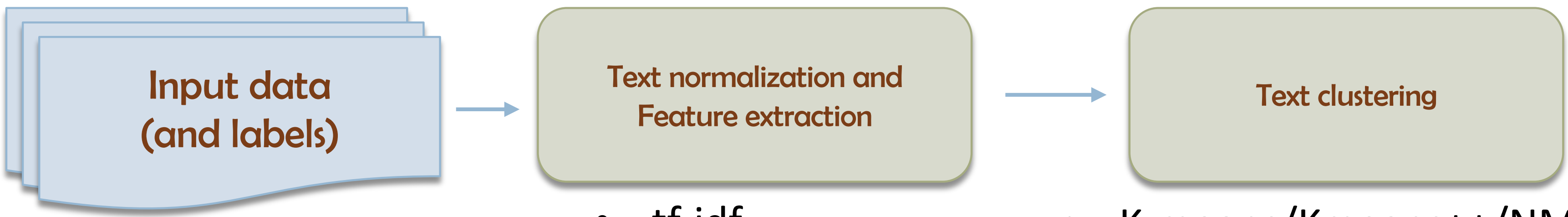
Text Clustering: Applications

❖ Concept Discovery and Retrieval from BBC news



http://www.nactem.ac.uk/bbc/

Text Clustering: Pipeline



- tf-idf
- word2vec

- K-means/Kmeans++/NMF
- Agglomerative Hierarchical Clustering

❖ Evaluation:
  - Rand Index
  - Purity
  - Normalized Mutual Information (NMI)
  - Silhouette Coefficient

Textual Data Analytics

I. Syntactical Analysis
   - Feature engineering
   - Representation learning

II. Semantic Analysis: turn textual data into high-quality information or actionable knowledge
   - Sentiment Analysis

Sentiment Analysis

- Computational study of opinions, sentiments, evaluations, attitudes, appraisal, affects, views, emotions, subjectivity, etc., expressed in text.
  - E.g. extract from text how people feel about different products (Reviews, blogs, discussions, news, comments, feedback, …)
- Is a review positive or negative toward the movie?

  - “Unbelievably disappointing”

  - “Full of zany characters and richly applied satire, and some great plot twists”

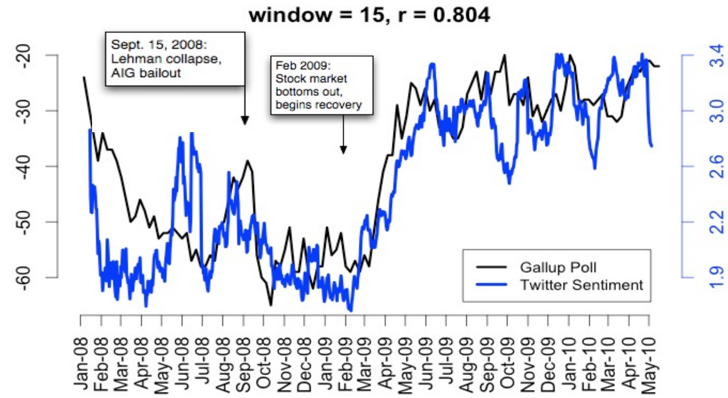
  - “This is the greatest screwball comedy ever filmed”

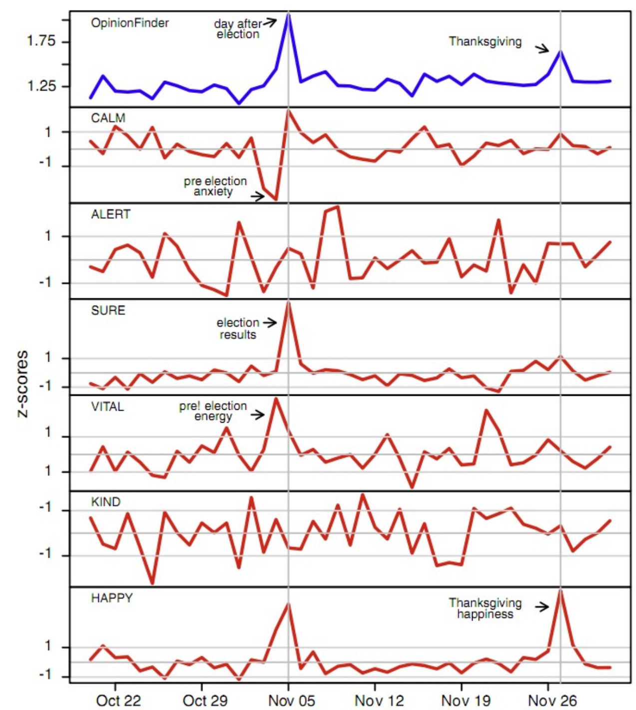
  - “It was pathetic. The worst part about it was the boxing scenes”

http://nlp.stanford.edu:8080/sentiment/rntnDemo.html

Sentiment Analysis: Applications

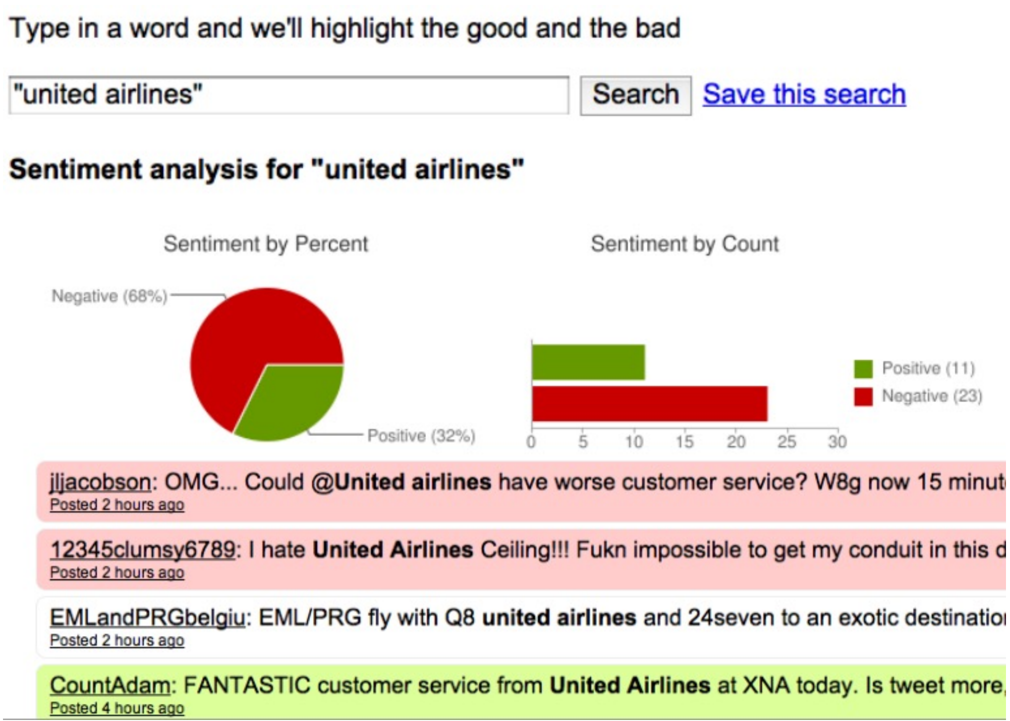
- Tracking sentiments toward topics over time:
  - **Predict public opinion**
    - E.g. Twitter sentiment versus Gallup Poll of Consumer Confidence
    - Brendan O'Connor, Ramnath Balasubramanyan, Bryan R. Routledge, and Noah A. Smith. 2010. From Tweets to Polls: Linking Text Sentiment to Public Opinion Time Series. In ICWSM-2010
  - **Predict stock market**
    - E.g. Johan Bollen, Huina Mao, Xiaojun Zeng. 2011. Twitter mood predicts the stock market, Journal of Computational Science 2:1, 1-8. 10.1016/j.jocs.2010.12.007.





Sentiment Analysis: Applications

- **For businesses and organizations:** product and service benchmarking, market intelligence.
  - Businesses spends a huge amount of money to find consumer sentiments and opinions. (Consultants, surveys, focus groups, etc.)



Twitter Sentiment App
Alec Go, Richa Bhayani, Lei Huang. 2009.
Twitter Sentiment Classification using Distant Supervision

Sentiment Analysis: Challenges

- **Sentiment can be tricky**
  - Honda Accords and Toyota Camrys are nice sedans
  - Honda Accords and Toyota Camrys are nice sedans, but hardly the best cars on the road
- **Subtlety**
  - Perfume review in Perfumes - the Guide:
    - "If you are reading this because it is your darling fragrance, please wear it at home exclusively, and tape the windows shut."
- **Polarity depends on domain context: e.g. "Soft"**
  - Positive in some domains (e.g., desserts)
  - Negative in other domains (e.g., building materials)
- **Thwarted Expectations and Ordering Effects:**
  - "This film should be brilliant. It sounds like a great plot, the actors are first grade, and the supporting cast is good as well, and Stallone is attempting to deliver a good performance. However, it **can't hold up**."
  - "Well as usual Keanu Reeves is nothing special, but surprisingly, the very talented Laurence Fishbourne is **not so good** either, I was surprised."

Types of Sentiment Analysis

- ❖ Depend on the level of requirements:
  1. **Simplest task:** Is the attitude of this text positive or negative?
  2. **More complex:** Rank the attitude of this text from 1 to 5
  3. **Advanced:** Detect the target

Sentiment Analysis: Simple Task

- Depend on the level of requirements:
  1. Simplest task: Is the attitude of this text positive or negative?
     - E.g. Is an IMDB movie review positive or negative?
       - http://www.cs.cornell.edu/people/pabo/movie-review-data
     - **Techniques:**
       - Classification approach
       - Lexicon approach

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up? Sentiment Classification using Machine Learning Techniques. EMNLP-2002, 79—86.
Bo Pang and Lillian Lee. 2004. A Sentimental Education: Sentiment Analysis Using Subjectivity Summarization Based on Minimum Cuts. ACL, 271-278

Sentiment Analysis: Classification Approach

- ❖ General pipeline
  1. Tokenization
  2. Feature Extraction
     - o Actual attributes of a product
     - o Indicative tokens: negation, emphasis, punctuation, etc.
  3. Compute the sentiment (e.g. positive/negative) using a classifier (e.g. SVM)

Sentiment Classification: Tokenization

- **Issues:**
  - Deal with HTML and XML markup
  - Twitter mark-up (names, hash tags)
  - Capitalization (preserve for words in all caps)
  - Phone numbers, dates
  - Emoticons
- **Solution:** regular expression
  - Useful code:
    - Brendan O'Connor twitter tokenizer
    - Christopher Potts sentiment tokenizer

Potts emoticons

```
[<>]?                          # optional hat/brow
[:;=8]                         # eyes
[\-o\*\']?                     # optional nose
[\)\]\(\[dDpP/\:\}\{@\|\\]     # mouth
|                              #### reverse orientation
[\)\]\(\[dDpP/\:\}\{@\|\\]     # mouth
[\-o\*\']?                     # optional nose
[:;=8]                         # eyes
[<>]?                          # optional hat/brow
```

Sentiment Classification: Feature Extraction

- **Which words to use?**
  - Only adjectives
  - Or all words

→ All words work better in some cases, you should try both in your dataset

- **Negation issue**
  - E.g. I didn't like this movie
  - Solution: add NOT_ to every word between negation and following punctuation

    → [NOT_like, NOT_this, NOT_movie]

Das, Sanjiv and Mike Chen. 2001. Yahoo! for Amazon: Extracting market sentiment from stock message boards. In Proceedings of the Asia Pacific Finance Association Annual Conference (APFA).
Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up? Sentiment Classification using Machine Learning Techniques. EMNLP-2002, 79—86.

Sentiment Analysis: Classification Approach

❖ General pipeline

1. Tokenization
2. Feature Extraction
3. **Compute the sentiment (e.g. positive/negative) using a classifier (e.g. SVM)**

Sentiment Analysis: Lexicon Approach

- Generalize pipeline:
  1. Tokenization (similar as above)
  2. Build or reuse a sentiment dictionary
  3. Compute the sentiment value for each token using the sentiment dictionary
  4. Aggregate the values (e.g. sum, max)

Lexicon Approach: Sentiment Dictionaries

- Bing Liu Opinion Lexicon
  - http://www.cs.uic.edu/~liub/FBS/sentiment-analysis.html
  - http://www.cs.uic.edu/~liub/FBS/opinion-lexicon-English.rar
  - 6786 words
    - 2006 positive
    - 4783 negative
- SentiWordNet
  - Home page: http://sentiwordnet.isti.cnr.it/
  - All WordNet synsets automatically annotated for degrees of positivity, negativity, and neutrality/objectiveness
    - [estimable(J,3)] "may be computed or estimated"

      Pos 0 Neg 0 Obj 1
    - [estimable(J,1)] "deserving of respect or high regard"

      Pos .75 Neg 0 Obj .25

Stefano Baccianella, Andrea Esuli, and Fabrizio Sebastiani. 2010 SENTIWORDNET 3.0: An Enhanced Lexical Resource for Sentiment Analysis and Opinion Mining. LREC-2010

Lexicon Approach: Building a dictionary

- What to do for domains where you don't have a lexicon?
  - Learn a lexicon (i.e. dictionary)
- Benefits:
  - Can be domain-specific
  - Can be more robust (more words)
- Bootstrapping technique:
  - Start with a seed set of words ('good', 'poor')
  - Find other words that have similar "meaning" (e.g. polarity):
    - Using "and" and "but"
    - Using words that occur nearby in the same document
    - Using WordNet synonyms and antonyms
    - Using word embeddings like word2vec
  - Use seeds and semi-supervised learning to induce lexicons

Vasileios Hatzivassiloglou and Kathleen R. McKeown. 1997. Predicting the Semantic Orientation of Adjectives. ACL, 174–181

Sentiment Analysis: Other Tasks (OPTIONAL)

- Finding aspect/attribute/target of sentiment
  - e.g. The restaurant has great food but awful service
  - Challenges:
    - The aspect name may not be in the sentence
    - Each domain has different aspects

M. Hu and B. Liu. 2004. Mining and summarizing customer reviews. In Proceedings of KDD.
S. Blair-Goldensohn, K. Hannan, R. McDonald, T. Neylon, G. Reis, and J. Reynar. 2008. Building a Sentiment Summarizer for Local Service Reviews. WWW Workshop.

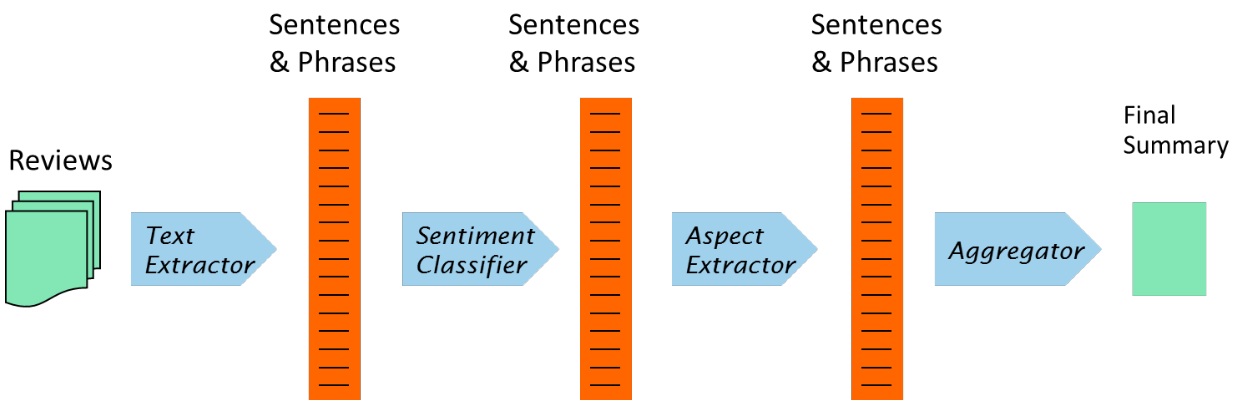
Sentiment Analysis: Other Tasks (OPTIONAL)

- Finding aspect/attribute/target of sentiment
  - e.g. The restaurant has great food but awful service
  - **Technique 1:** frequent phrases + rules
    - Find all highly frequent phrases/terms across reviews (“food”)
    - Filter by rules like “occurs right after sentiment word”
      - “…great food” means “food” a likely aspect
  - **Technique 2:** supervised classification
    - Hand-label a small corpus of restaurant review sentences with aspect
    - Train a classifier to assign an aspect to a sentence
    - e.g. labels = food, décor, service, value, or NONE

M. Hu and B. Liu. 2004. Mining and summarizing customer reviews. In Proceedings of KDD.
S. Blair-Goldensohn, K. Hannan, R. McDonald, T. Neylon, G. Reis, and J. Reynar. 2008. Building a Sentiment Summarizer for Local Service Reviews. WWW Workshop.

Sentiment Analysis: Put it altogether (OPTIONAL)

- Put it altogether



Final summary

Rooms (3/5 stars, 41 comments)

(+) The room was clean and everything worked fine – even the water pressure ...

(+) We went because of the free room and was pleasantly pleased ...

(-) ...the worst hotel I had ever stayed at ...

Service (3/5 stars, 31 comments)

(+) Upon checking out another couple was checking early due to a problem ...

(+) Every single hotel staff member treated us great and answered every ...

(-) The food is cold and the service gives new meaning to SLOW.

Dining (3/5 stars, 18 comments)

(+) our favorite place to stay in biloxi.the food is great also the service ...

(+) Offer of free buffet for joining the Play

S. Blair-Goldensohn, K. Hannan, R. McDonald, T. Neylon, G. Reis, and J. Reynar. 2008. Building a Sentiment Summarizer for Local Service Reviews. WWW Workshop

References

[1] https://youtu.be/buoTFL3mxJw

[2] Probabilistic Latent Semantic Analysis, Hofmann T., Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence,1999 [PLSA99]

[3] Unsupervised Learning by Probabilistic Latent Semantic Analysis, Hofmann, T., Machine learning 42.1 (2001): 177-196. [PLSA01]

[4] Latent Dirichlet allocation, D. M. Blei, A. Y. Ng, and M. I. Jordan, Journal of Machine Learning Research, 3(5):993–1022, 2003. [LDA03]

[5] On an equivalence between PLSI and LDA. Girolami, M. & Kabán, A., Proceedings of the 26th annual international ACM SIGIR conference on Research and development in information, 2003 .

[6] Steinberger, Josef, and Karel Jezek. "Using latent semantic analysis in text summarization and summary evaluation." Proc. ISIM'04. 2004.

[7] Gong, Yihong, and Xin Liu. "Generic text summarization using relevance measure and latent semantic analysis." Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval. ACM, 2001.

[8] https://research.googleblog.com/2016/08/text-summarization-with-tensorflow.html

[9] https://github.com/tensorflow/models/tree/master/textsum

[10] https://hackernoon.com/how-to-run-text-summarization-with-tensorflow-d4472587602d

[11] https://ronxin.github.io/wevi/

[12] https://blog.acolyer.org/2016/04/21/the-amazing-power-of-word-vectors/

[13] https://web.stanford.edu/class/cs124/

[14] https://www.quora.com/What-are-the-continuous-bag-of-words-and-skip-gram-architectures