TEXT MINING

- Text is everywhere.
  - Twitter
  - Blogs
  - Articles
  - Emails
  - Books
  - surveys
  - Etc
- What do we do with it?
- Unstructured: How do we represent it?

TEXT MINING

- Application Topics:
  - Information Retrieval: retrieve information
  - Search: Search engines
  - Classification: Categorize documents
  - Clustering: Organize
  - Topic Modelling: Find the topic, theme of text
  - Sentiment Analysis: What is the sentiment
  - POS: Parts of speech tagging
  - Record Linkage: Link data sources
  - Speech to text
  - Machine Translation

TEXT MINING

- How to represent text is a difficulty
  - Any ideas?

TEXT MINING

- How to represent text is a difficulty
  - Any ideas?
- Bag of Words, Vector Space

TEXT MINING

- How to represent text is a difficulty
  - Any ideas?
- Bag of Words, Vector Space
  - Tokenize text into “words”

TEXT MINING

- How to represent text is a difficulty
  - Any ideas?
- Bag of Words, Vector Space
  - Tokenize text into “words”
  - Treat each token as a feature

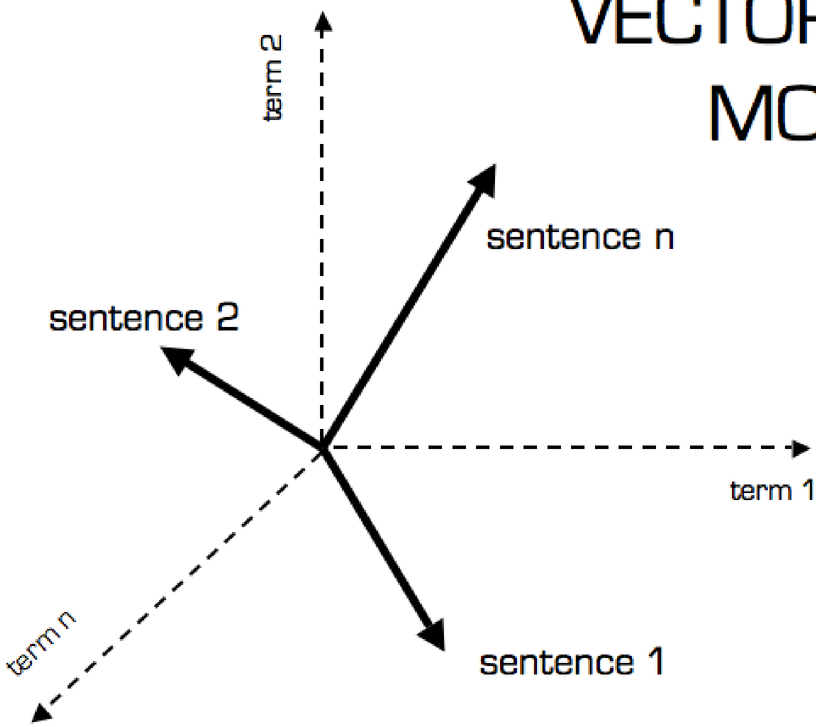
TEXT MINING

- How to represent text is a difficulty
  - Any ideas?
- Bag of Words, Vector Space
  - Tokenize text into “words”
  - Treat each token as a feature
  - Value for feature:
    - Binary if present in document
    - Token frequency
    - TF-IDF
    - etc

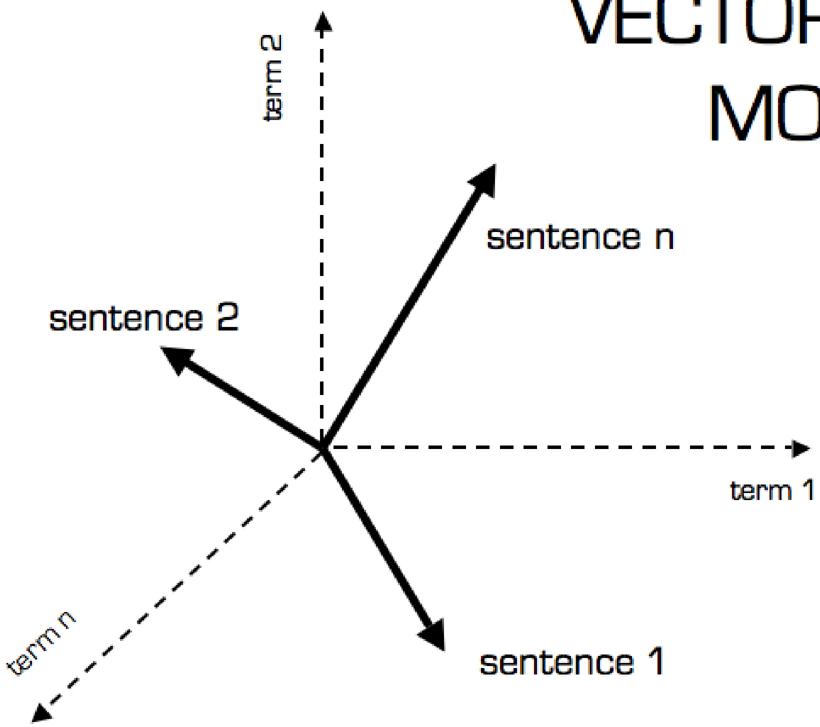
TEXT MINING

- How to represent text is a difficulty
  - Any ideas?
- Bag of Words, Vector Space
  - Tokenize text into “words”
  - Treat each token as a feature
  - Value for feature:
    - Binary if present in document
    - Token frequency
    - TF-IDF
    - etc
  - Each document can be represented as a vector

VECTOR SPACE MODEL



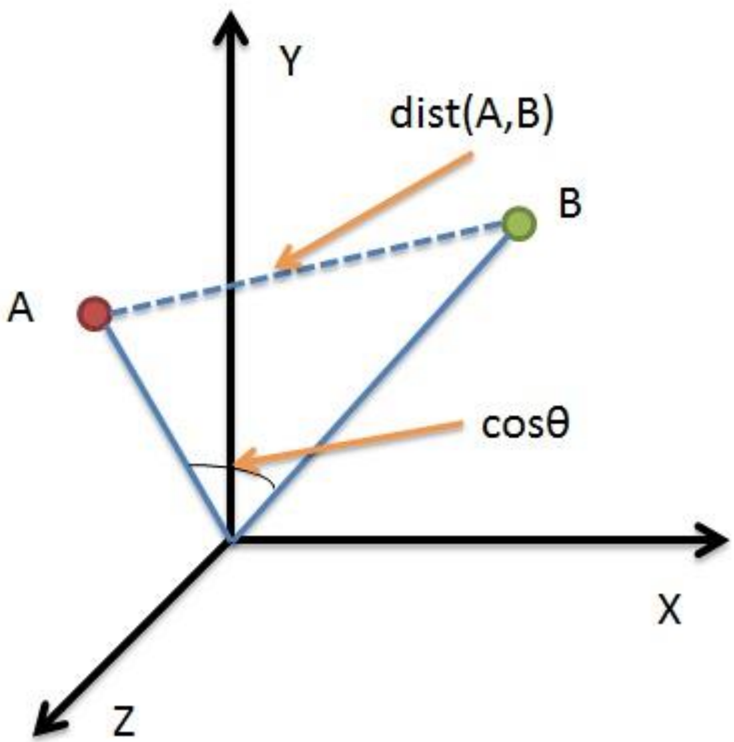
VECTOR SPACE MODEL

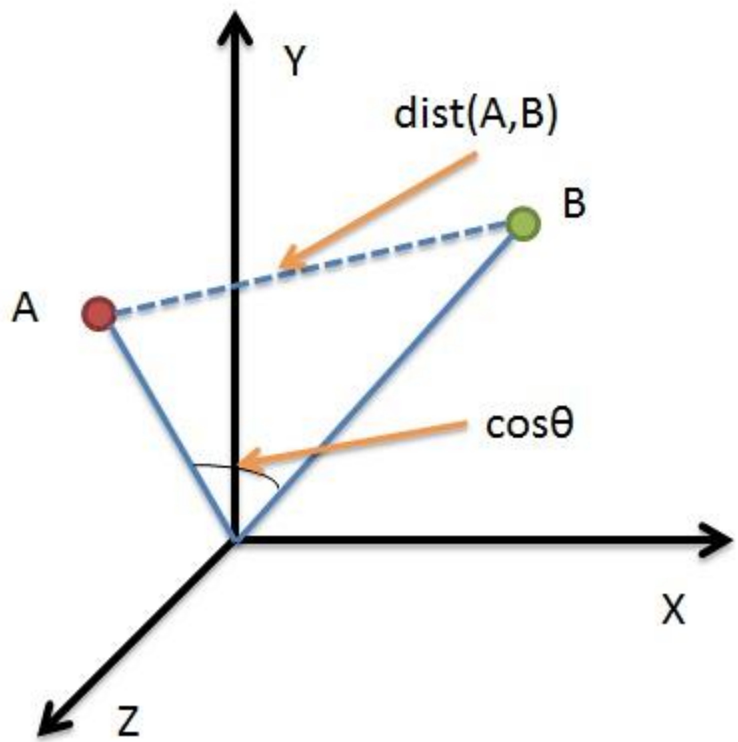


Document:

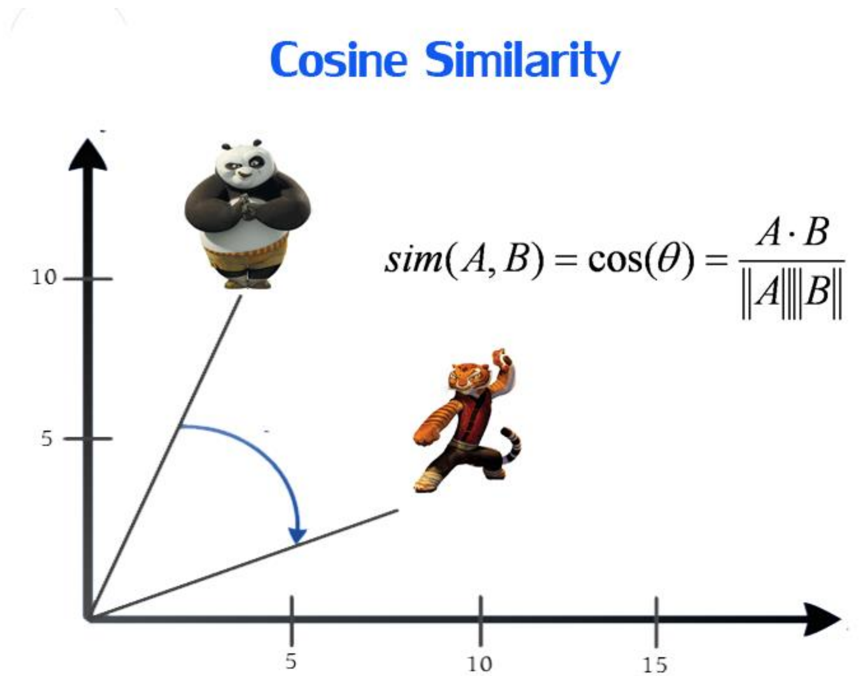
- Classification
- Clustering
  - Distance
  - Cosine similarity

$$similarity = \cos(\theta) = \frac{A \cdot B}{\|A\| \|B\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}}$$


Y
dist(A,B)
B
A
cosθ
X
Z



Y
dist(A,B)
B
A
cosθ
X
Z



Cosine Similarity
$sim(A,B) = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|}$
10
5
5
10
15

- Observations:
  - High dimensional: Lot’s of tokens, characters, etc
  - Sparse: Most documents have a small number of tokens
  - Common words: The, I, and, etc
  - Word similarity: grocery, groceries
  - Word ordering: Only considers presence of words not ordering
    - Is this the first document
    - This is the first document
  - Misspelling, Acronyms, Ambiguity, etc

TEXT MINING

- N-gram: ordered sequence of n-terms
  - N = 1 unigram: Single token
  - N = 2 bigram: two ordered tokens

TEXT MINING

- N-gram: ordered sequence of n-terms
  - N = 1 unigram: Single token
  - N = 2 bigram: two ordered tokens
- Example: This is a sentence
  - Unigrams: {this, is, a, sentence}
  - Bigrams: {this is, is a, a sentence}

TEXT MINING

- N-gram: ordered sequence of n-terms
  - N = 1 unigram: Single token
  - N = 2 bigram: two ordered tokens
- Example: This is a sentence
  - Unigrams: {this, is, a, sentence}
  - Bigrams: {this is, is a, a sentence}
- Stop words: Common words. Usually removed

TEXT MINING

- N-gram: ordered sequence of n-terms
  - N = 1 unigram: Single token
  - N = 2 bigram: two ordered tokens
- Example: This is a sentence
  - Unigrams: {this, is, a, sentence}
  - Bigrams: {this is, is a, a sentence}
- Stop words: Common words. Usually removed
- Stemming: Reduce word to its stem, root
  - Stem, stemming, stemmer, stemmed -> stem

TF-IDF: Term Frequency-Inverse Document Frequency

TF-IDF: Term Frequency-Inverse Document Frequency

- Term Frequency TD(t,d): Frequency of term in document

TEXT MINING

TF-IDF: Term Frequency-Inverse Document Frequency

- Term Frequency TD(t,d): Frequency of term in document
- Inverse Document Frequency IDF(t):
  - log(# documents/# of documents containing t)

TEXT MINING

TF-IDF: Term Frequency-Inverse Document Frequency

- Term Frequency TD(t,d): Frequency of term in document
- Inverse Document Frequency IDF(t):
  - log(# documents/# of documents containing t)
- TF-IDF(d,t) = TF*IDF

TEXT MINING

TF-IDF: Term Frequency-Inverse Document Frequency

- Term Frequency TD(t,d): Frequency of term in document
- Inverse Document Frequency IDF(t):
  - log(# documents/# of documents containing t)
- TF-IDF(d,t) = TF*IDF
- If a term is frequently appearing in a document TF goes up

TEXT MINING

TF-IDF: Term Frequency-Inverse Document Frequency

- Term Frequency TD(t,d): Frequency of term in document
- Inverse Document Frequency IDF(t):
  - log(# documents/# of documents containing t)
- TF-IDF(d,t) = TF*IDF
- If a term is frequently appearing in a document TF goes up
- What about words like “the” which appear very frequently?

TEXT MINING

TF-IDF: Term Frequency-Inverse Document Frequency

- Term Frequency TD(t,d): Frequency of term in document
- Inverse Document Frequency IDF(t):
  - log(# documents/# of documents containing t)
- TF-IDF(d,t) = TF*IDF
- If a term is frequently appearing in a document TF goes up
- What about words like “the” which appear very frequently?
  - Common words usually appear in multiple documents

TEXT MINING

TF-IDF: Term Frequency-Inverse Document Frequency

- Term Frequency TD(t,d): Frequency of term in document
- Inverse Document Frequency IDF(t):
  - log(# documents/# of documents containing t)
- TF-IDF(d,t) = TF*IDF
- If a term is frequently appearing in a document TF goes up
- What about words like “the” which appear very frequently?
  - Common words usually appear in multiple documents
  - The more common the word, more documents it’s in, the closer (# of documents containing t) gets to (# documents)

TEXT MINING

TF-IDF: Term Frequency-Inverse Document Frequency

- Term Frequency TD(t,d): Frequency of term in document
- Inverse Document Frequency IDF(t):
  - log(# documents/# of documents containing t)
- TF-IDF(d,t) = TF*IDF
- If a term is frequently appearing in a document TF goes up
- What about words like “the” which appear very frequently?
  - Common words usually appear in multiple documents
  - The more common the word, more documents it’s in, the closer (# of documents containing t) gets to (# documents)
  - So the closer the ratio goes to 1

TEXT MINING

TF-IDF: Term Frequency-Inverse Document Frequency

- Term Frequency TD(t,d): Frequency of term in document
- Inverse Document Frequency IDF(t):
  - log(# documents/# of documents containing t)
- TF-IDF(d,t) = TF*IDF
- If a term is frequently appearing in a document TF goes up
- What about words like “the” which appear very frequently?
  - Common words usually appear in multiple documents
  - The more common the word, more documents it’s in, the closer (# of documents containing t) gets to (# documents)
  - So the closer the ratio goes to 1
  - Log(1) = 0

TEXT MINING

TF-IDF: Term Frequency-Inverse Document Frequency

- Term Frequency TD(t,d): Frequency of term in document
- Inverse Document Frequency IDF(t):
  - log(# documents/# of documents containing t)
- TF-IDF(d,t) = TF*IDF
- If a term is frequently appearing in a document TF goes up
- What about words like “the” which appear very frequently?
  - Common words usually appear in multiple documents
  - The more common the word, more documents it’s in, the closer (# of documents containing t) gets to (# documents)
  - So the closer the ratio goes to 1
  - Log(1) = 0
  - So common words IDF value goes to 0 thus TF-IDF gets small

Topic Modelling: Finding the topics, themes occurring in a collection of documents

- LDA: Latent Dirichlet Allocation
  - Blei, Ng, Jordan 2003
  - Each document assumed to be a mixture of topics
  - Probability of document to topic
  - Words associated with topics
- Matrix Factorization
- Etc

Useful Python Tools:

- NLTK
- Sklearn
- LDA