# Natural Language Processing

## Lecture #3
## Document Retrieval and Classification

Hutchatai Chanlekha

Department of Computer Engineering, Kasetsart University

## Bag of Words

- Most IR and document classification systems assume meaning of documents resides solely in the words that are contained within them.
  - Ignore syntactic information
  - These approaches are often referred to as "***bag of words***" methods

## Terminology in IR

- In IR
  - Document
    - refers generically to the unit of text indexed in the system and available for retrieval
    - A document can refer to: News paper articles, Encyclopedia entries, Paragraphs, Sentences, Web page, etc.
  - Collection
    - A set of documents being used to satisfy user requests
  - Term
    - Lexical item that occurs in a collection, but it may also include phrases
  - Query
    - A user's information need, expressed as a set of terms (could be set of terms/phrases or query documents)

Here, we will focus on using document as a query

## Evaluation Metric

- Precision
  - Precision = $\frac{\text{\# of relevant documents returned}}{\text{\# of documents returned}}$
- Recall
  - Recall = $\frac{\text{\# of relevant documents returned}}{\text{total \# of relevant documents in the collection}}$
- F-measure
  - F-score = $\frac{2*P*R}{P+R}$

# Vector Space Model

- Documents and queries are represented as vectors of features representing the terms that occur within the collection
  - Value of each features indicating the presence or absence of a given term in a given document.
  - Vectors can be represented as follows:

Document: $\vec{d}_j = (t_{1,j}, t_{2,j}, t_{3,j}, \ldots, t_{N,j})$

Query: $\vec{q}_k = (t_{1,k}, t_{2,k}, t_{3,k}, \ldots, t_{N,k})$

*t* features represent N terms that occur in the collection

$$\begin{array}{ccc} & & \text{Doc}_1 \\ & w_1 & \cdots \\ & w_2 & \cdots \\ \text{vocab} & \cdots & \\ & w_N & \cdots \end{array}$$

N = # of terms in a collection
- All terms
- Select only some terms
  - such as keep only words with *freq*. > *t*

## Feature Vector Representation

- Strategy I
  - Features take on the value of one or zero
    - $t_i = 1 \rightarrow$ term *i* presents in the document
      $t_i = 0 \rightarrow$ term *i* absents from the document
  - Given this approach
    - Simple way to determine the relevance of a document to a query is to determine the number of terms they have in common.
    - Similarity metric

$$sim(\vec{q}_k, \vec{d}_j) = \sum_{i=1}^{N} t_{i,k} \times t_{i,j}$$

  - Problem
    - Fails to capture the fact that some terms are more important to the meaning of a document than others
    - To solve this
      - Replace one-zero scheme with numerical weights that indicate the importance of the terms

# Feature Vector Representation (cont.)

- Strategy II
  - Charaterize documents as vectors of term weights
  - Features take on the weighted values indicating the importance of the terms

$$\vec{d}_j = (w_{1,j}, w_{2,j}, w_{3,j}, \ldots, w_{N,j})$$

$$\vec{q}_k = (w_{1,k}, w_{2,k}, w_{3,k}, \ldots, w_{N,k})$$

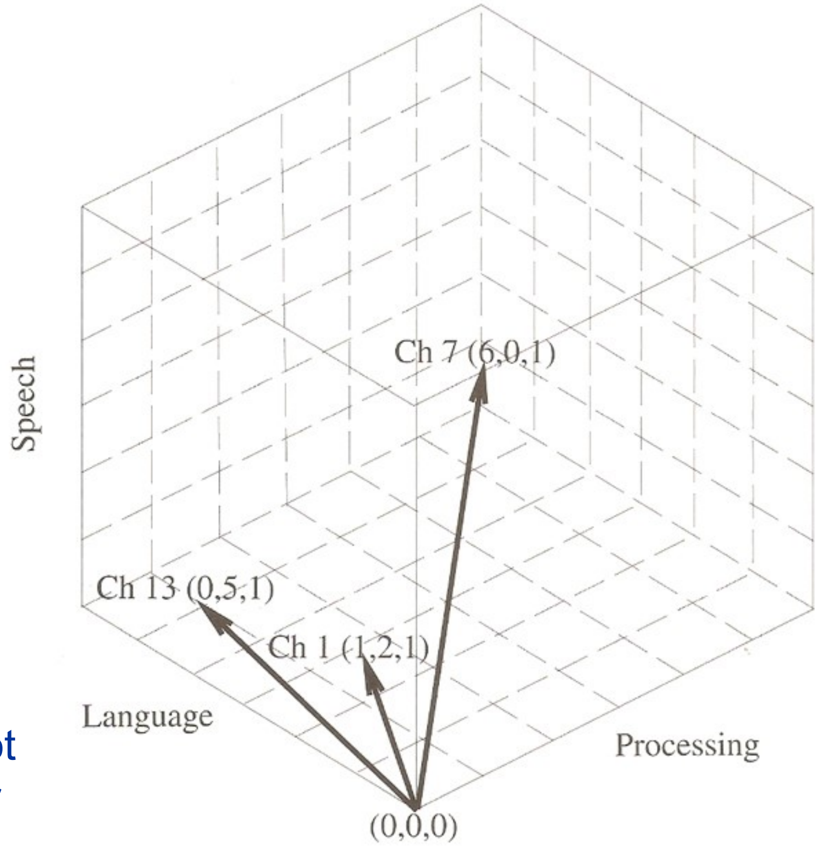
$$\begin{array}{ccccc} & \mathrm{Doc}_1 & \mathrm{Doc}_2 & .... & \mathrm{Doc}_n \\ w_1 & \cdots & \cdots & \cdots & \cdots \\ w_2 & \cdots & \cdots & \cdots & \cdots \\ \vdots & & & & \\ w_N & \cdots & \cdots & \cdots & \cdots \end{array}$$

    - $w_{i,j}$ represents the weight of term $i$ in document $j$

  - View the features as dimensions in a multi-dimensional space.
  - Weights (i.e. feature values) serve to locate documents in that space
  - If the query is translated into a vector, documents that are located close to the query can be judged as being more relevant than documents that are further away.

# Feature Vector Representation (cont.)

- Example:
  - Suppose raw term frequency is used in document as a weight
    - Simplify space consisting of three dimensions corresponding to the terms *"speech", "language", "processing"*



But using absolute frequency values may not be appropriate.

In Ch.7, the importance freq. of "speech" is not about the value, but rather about the contributor to the meaning of the document

In Ch.1, the importance is not about the specific value (1,2,1), but rather about the fact that the three dimensions have roughly similar values

# Feature Vector Representation (cont.)

- Using absolute values, such as frequency count, may not be appropriate
- Too much emphasis on the absolute values of coordinates of each document

- Normalizing document vectors
  - Converting all vectors to a standard length. → divided by $\sqrt{\sum_{i=1}^{N} w_i^2}$
  - Unit length: dividing each dimension by the overall length of the vector
  - Normalizing eliminates the importance of the exact length of the document’s vector and emphasize the direction of the vector instead

## Similarity Metric

Cosine similarity $$sim(\vec{q}_k, \vec{d}_j) = \vec{q}_k \cdot \vec{d}_j = \sum_{i=1}^{N} w_{i,k} \times w_{i,j}$$

- Dot product between normalizing vectors (i.e. unit vectors)
  - No effect from magnitudes
- Compute cosine
  - Identical → cosine = 1
  - No common terms → orthogonal → cosine = 0

## Term Weighting

- Method used to assigned terms weights has an enormous impact
- Two critical factors in deriving effective term weights
  - Term frequency
    - Terms which occur frequently may reflect its meaning more strongly than terms that occur less frequently
    - Thus, should have higher weights
  - Inverse document frequency
    - Consider distribution of terms across the collection
    - Terms that are limited to a documents are useful for discriminating these documents from the rest
      - More frequent terms are less useful
    - Measure that favors terms which occur in fewer documents:

      $$idf_i = \log\left(\frac{N}{df(w_i)}\right)$$

      - $N$ = # of docs in collection; $df(w_i)$ = # of docs that contain $w_i$
- Combine the tf and idf
  - tf·idf weighting: $w_{i,j} = tf_{i,j} \times idf_i$

## Term Weighting (cont.)

- Query by using document can use the same method for query vectorization
- However, weighting for short-text query…
  - User queries are not very much like documents
  - Usually short
  - Raw term frequency in query is not likely to be a very useful factor
  - Many works proposed different term weighting for short-text
    - Salton and Buckley (1988) recommend formula for weighting query terms

$$w_{i,k} = \left(0.5 + \frac{0.5tf_{i,k}}{\max_j tf_{j,k}}\right) \times idf_i$$

$\max_j tf_{j,k}$ denotes the frequency of the most frequent term in document k

## Other weighting scheme

- TF-IDF-CF
- TFC
- LTC
- Ref: http://www.ipcsit.com/vol47/009-ICCTS2012-T049.pdf

## Term Selection and Criterion

- Words are used to index the documents in the collection
- Two common variation
  - Stemming/Lemmatization
  - Stop list

# Stemming/Lemmatization

- Morphological variants of a lexical item should be …
  - listed separately, or
  - collapsed into a single root form
- Example:
  - process, processing, processed, processes

| Terms | Freq. |
|---|---|
| process | 17 |
| processing | 12 |
| processed | 5 |
| processes | 9 |

| Terms | Freq. |
|---|---|
| process | 43 |

- Problem
  - Throw away useful distinctions
    - Increase recall: return all docs with terms that are morphologically related
    - May decrease precision: return semantically unrelated

# Stop list

- High frequency words that are eliminated from the representation of both documents and queries
- High frequency, close class terms are seen as carrying little semantic weight
  - Such as, “and”, “or”, “but”, “to”, …
- Downside
  - Difficult to search for phrases that contain words in the stop list
  - Example: “*to be or not to be*” $\rightarrow$ “*not*”

## Document Classification

- Text classification, Text categorization, Document classification, Document categorization
- Classify text into a predefined set of classes
- Approaches
  - Centroid-based algorithm
  - Machine Learning
    - Many classification-based techniques, such as SVM, Naïve Bayes, etc.

## Centroid Algorithm

- Simple algorithm
- For each category, one centroid is created
  - Let $S^+$ denote a set of documents in positive class
  - Centroid of the category is:

$$c = \frac{1}{|S^+|} \sum_{d \in S^+} \vec{d}$$

    - Vector representation of the documents could be cosine-normalized *tfidf* vector.
- Similarity measure: $$sim(c,d) = \frac{c \cdot d}{\|c\|_2 \|d\|_2}$$
    - If *d* and *c* is already normalized, then the division is not necessary.
- Classification method
  - Defined threshold *t*; *d* belongs to the category iff sim(*c*,*d*) $\geq t$
  - Or assign *d* to the category whose centroid has the highest similarity to *d*.

## Machine Learning for Classification

- Classification-based Machine learning can be used for document classification
  - Such as SVM, k-NN, Decision Tree, Random Forest, ANN, etc.
  - General approach
    - Convert document to vector representation using bag-of-word model
    - Target attribute (answer) is the class of each document
    - Training with ML technique
  - Another ML technique for document classification: Naïve Bayes

# Naïve Bayes Classifier

- Assume target function $f : X \rightarrow V$, where each instance $x$ described by attributes $\langle a_1, a_2, \ldots, a_n \rangle$
- Most probable value of $f(x)$ is:

$$v_{MAP} = \underset{v_j \in V}{\arg\max}\, P(v_j \mid a_1, a_2, \ldots, a_n)$$

$$v_{MAP} = \underset{v_j \in V}{\arg\max} \frac{P(a_1, a_2, \ldots, a_n \mid v_j) P(v_j)}{P(a_1, a_2, \ldots, a_n)}$$

$$= \underset{v_j \in V}{\arg\max}\, P(a_1, a_2, \ldots, a_n \mid v_j) P(v_j)$$

- Naïve Bayes assumption:

$$P(a_1, a_2, \ldots, a_n \mid v_j) = \prod_i P(a_i \mid v_j)$$

which gives

**Naïve Bayes classifier**: $v_{NB} = \underset{v_j \in V}{\arg\max}\, P(v_j) \prod_i P(a_i \mid v_j)$

## Naïve Bayes Algorithm

- Naïve_Bayes_Learn(*examples*)

```
For each target value $v_j$
    $\hat{P}(v_j) \leftarrow$ estimate $P(v_j)$

For each attribute value $a_i$ of each attribute a
        $\hat{P}(a_i|v_j) \leftarrow$ estimate $P(a_i|v_j)$
```

- Classify_New_Instance(*x*)

$$v_{NB} = \arg\max_{v_j \in V} \hat{P}(v_j) \prod_{a_i \in x} \hat{P}(a_i \mid v_j)$$

# Problem with Naïve Bayes

- What if none of the training instances with target value $v_j$ have attribute value $a_i$?

In this case: $\hat{P}(a_i \mid v_j) = 0$ and

$$\hat{P}(v_j)\prod_i \hat{P}(a_i \mid v_j) = 0$$

Typical solution is Bayesian estimate for $\hat{P}(a_i | v_j)$

$$\hat{P}(a_i \mid v_j) \leftarrow \frac{n_c + mp}{n + m}$$

where

- $n$ is the number of training examples for which $v = v_j$
- $n_c$ is the number of examples for which $v = v_j$ and $a = a_i$
- $p$ is prior estimate for $\hat{P}(a_i | v_j)$
- $m$ is weight given to prior (i.e. number of "virtual" examples)

# Document Classification Methodology

- *LEARN_NAIVE_BAYES_TEXT* (*Examples*, *V*)
  1. Collect all words and other tokens that occur in Examples
     - Vocabulary $\leftarrow$ all distinct words and other tokens in Examples
  2. Calculate the required $P(v_j)$ and $P(w_k|v_j)$ probability terms
     - For each target value $v_j$ in $V$ do
       - $docs_j$ $\leftarrow$ subset of *Examples* for which the target value is *vj*
       - $P(v_j) \leftarrow \dfrac{|docs_j|}{|Examples|}$
       - $Text_j$ $\leftarrow$ a single document created by concatenating all members of $docs_j$
       - $n$ $\leftarrow$ total number of words in $Text_j$ (counting duplicate words multiple times)
       - For each words $w_k$ in *Vocabulary*
         - $n_k$ $\leftarrow$ number of times word wk occurs in $Text_j$
         - $P(w_k | v_j) \leftarrow \dfrac{n_k + 1}{n + |Vocabulary|}$

## Document Classification Methodology (cont.)

- *CLASSIFY_NAIVE_BAYES_TEXT* (Doc)
  - Return $v_{NB}$, where

$$v_{NB} = \underset{v_j \in V}{\operatorname{argmax}} P(v_j) \prod_{w_i \in DOCs} P(w_i | v_j)$$

## Precision Recall trade-off

- Precision and recall usually trade off against each other.
- If we seek relatively high precision, we may need to give up on recall.
  - For example: by selecting different threshold for classification

| | |
|---|---|
| + | 0.9 |
| + | 0.8 |
| + | 0.65 |
| + | 0.7 |
| + | 0.58 |
| - | 0.53 |
| - | 0.4 |
| - | 0.2 |

t = 0.8
t = 0.7
t = 0.6
t= 0.5

## Summary

- Corpus preprocessing for document classification/IR
  - Tokenization
  - Lemmatization
  - Stop word removal
  - Low-frequency word removal
- Generate vocabulary
- Term weighting computation
  - Calculate term weighting according to the selected term weighting scheme
- Generate document vector for each document
- Train classification model by using the generated document vectors

## Workshop

- Document Classification
  - 4-5 persons per group
  - Instruction and dataset will be posted in google classroom