Text data analysis and applications

Phan Xuân Hiểu
Bài giảng của DSLab
Viện nghiên cứu cao cấp về Toán (VIASM)
Contents

- **Part 1:** Overview of text data analysis
- **Part 2:** Preprocessing text data and text representation
- **Part 3:** Text classification, clustering, and topic analysis
- **Part 4:** Word and phrase analysis
- **Part 5:** Information extraction from texts
- **Part 6:** Vietnamese text and language processing
- **Part 7:** Text analysis application topics
Contents

- **Part 1**: Overview of text data analysis
- **Part 2**: Preprocessing text data and text representation
- **Part 3**: Text classification, clustering, and topic analysis
- **Part 4**: Word and phrase analysis
- **Part 5**: Information extraction from texts
- **Part 6**: Vietnamese text and language processing
- **Part 7**: Text analysis application topics
What is text data analysis?

Text data analysis is the process of deriving relevant and useful information or knowledge from text data.

- **Major topics or problems in text analysis:**
  - Text preprocessing and representation
  - Text retrieval and search
  - Text classification and clustering
  - Word and phrase analysis
  - Information extraction from texts

- **Common methods and techniques:**
  - Rule-based methods
  - Statistical machine learning methods
Text data analysis and related areas

- Text collections
- Word, phrase analysis
- Classification and filtering
- Clustering
- Topic analysis
- Shallow parsing
- Syntactic analysis
- Information extraction

Text Mining

- Corpora
- Lexical analysis
- Language models
- Disambiguation
- Shallow parsing
- Syntactic analysis
- Semantic analysis
- Pragmatic analysis
- Discourse analysis
- Inference
- Language understanding
- Language generation
- Computational linguistics

NLP

- Text crawling and indexing
- Query analysis
- IR models
- Language models
- Relevance feedback

Information Retrieval

- Text Data Analysis
Views from text retrieval and text mining
View from text information system

- Retrieval applications
  - Summarization
  - Visualization
  - Filtering
  - Clustering
  - Search
  - Extraction
  - Categorization
  - Topic analysis

- Information access

- Natural language content analysis

- Mining applications
  - Knowledge acquisition

Text
View from natural language processing and understanding
Why text data analysis?

- A wide range of applications:
  - Smart text information management
  - Opinion mining and sentiment analysis
  - Social media listening
  - Content matching for contextual advertising
  - Comparison and market search (IE)
  - Text summarization applications
  - Text generation applications
  - Natural language search
  - Question-answering and chatbots
  - Text analysis applications in airlines

Source: ISD - 2014, Structured Data vs. Unstructured Data: The Balance of Power Continues to Shift
Text data analysis tools and platforms

- **Scikit-learn** (Python/Cython):
  - APIs for various machine learning methods and techniques (classification, regression, clustering etc.)
  - Built on top of Cython (including high-performance C libraries: LAPACK, LibSVM, Boost)
  - Open source and commercially usable

- **NLTK** (Natural Language Tool-Kit - Python):
  - Corpora, lexical resources, grammars, pre-trained models, NLP algorithms

- **Gensim** (topic modeling for humans - Python):
  - Topic modeling, text similarity, word2vec, etc.

- **spaCy** (Python and Cython):
  - Industrial-strength language processing, deep learning and large-scale applications

- **NetworkX** (Python):
  - A comprehensive graph analytics package.

- **Yellowbrick** (Python):
  - A suite of visual diagnostic tools for the analysis and interpretation of machine learning workflows
Contents

- Part 1: Overview of text data analysis
- Part 2: Preprocessing text data and text representation
- Part 3: Text classification, clustering, and topic analysis
- Part 4: Word and phrase analysis
- Part 5: Information extraction from texts
- Part 6: Vietnamese text and language processing
- Part 7: Text analysis application topics
Part 2: Preprocessing text data and text representation

- Text document and collection
- Text tokenization and stemming
- Word and term distribution
- Stop word and stop word removal
  - Stop words in aviation domain?
- Text vectorization techniques
  - Bag of words
  - N-grams and bag of n-grams
  - Distributed representation
Text document and collection

- Types of text document
  - Newswire, online news articles
  - Government documents
  - Scientific articles
  - Books
  - Email messages
  - Social media posts (Blog, Facebook, Twitter, etc.)
  - Instant/chat messages
  - A search query
  - Feedback/comments (e.g., airline passengers)

- Text collection
  - A set of documents
  - Also known as “text dataset” or “text corpus”
Preprocessing raw text data

How to Manage Stress Like an Olympic Biathlete

PYEONGCHANG, South Korea — Race across the snow on skis as fast as you can. Now stop and shoot a target the size of an Oreo about 54 yards away. If you miss, you’ll ski penalty laps before you are allowed to race to the next set of targets.

Most of us will never try the biathlon, a uniquely stressful sport that demands both physical intensity and emotional calm. But that doesn’t mean we can’t learn from it. Talking to an Olympic biathlete about how she trains for competition can offer a life lesson in managing stress and dialing back intensity and aggression in an instant.
A small airline passenger feedback text collection

- Number of feedback comments (documents): 601
- The total number of word/term occurrences: 27703 (including punctuations)
- Number of unique words/tokens: 3417
- The total number of sentences: 1546
- The average document length (number of words): 46
- The average number of sentences per comment: 2.57
- The average sentence length: 17.9 (including words and punctuations)
Inform the staff on the routes followed by our luggage. My Paris-Siem Reap trip consisted of 3 Vietnam Airlines flights (2 stops: Hanoi then Ho Chi Minh city). I buy the full journey at the same time on Opodo (1 trip only and not 3 trips). In Paris, I was told that my luggage would be sent to the final destination, in fact I had to get it at each stop. It sounds big but it's the reality. The reasons given: Stopover 1: domestic flight Hanoi-Ho Chi Minh, I got my bag on the treadmill and I had to drop it 10 meters away. My bag and I made the trip separately to the domestic airport. Stop 2: Flight Vietnam Airlines announced, in fact operated by Cambodian. I had to re-register. Even at stopovers, the information given proved to be wrong. In Hanoi, I was also told that my bag would be sent to the final destination (Siem Reap) but suspicious I asked the question in Ho Chi Minh, where I was told to pick up my bag.

```python
from nltk import sent_tokenize
for sentence in sent_tokenize(paragraph):
    print(sentence)
```
1. Inform the staff on the routes followed by our luggage.
3. I buy the full journey at the same time on Opodo (1 trip only and not 3 trips).
4. In Paris, I was told that my luggage would be sent to the final destination, in fact I had to get it at each stop.
5. It sounds big but it's the reality.
6. The reasons given: Stopover 1: domestic flight Hanoi-Ho Chi Minh, I got my bag on the treadmill and I had to drop it 10 meters away.
7. My bag and I made the trip separately to the domestic airport.
9. I had to re-register.
10. Even at stopovers, the information given proved to be wrong.
11. In Hanoi, I was also told that my bag would be sent to the final destination (Siem Reap) but suspicious I asked the question in Ho Chi Minh, where I was told to pick up my bag.
Word tokenization

sents = ["The flight crew should personalize the contact with the passenger, that is to say when ordering the meal and come individually to thank each passenger, as is done at other airlines (Etihad for example).", "The staff should also be more present between the two snacks to meet the potential demands of passengers It surprised me that boarding nobody asks me to present my visa, knowing that in case of refusal during passport control the return is at the expense of the airline ..."]

from nltk import word_tokenize
for sent in sents:
    print(word_tokenize(sent))

["The", "flight", "crew", "should", "personalize", "the", "contact", "with", "the", "passenger", ",", "that", "is", "to", "say", "when", "ordering", "the", "meal", "and", "come", "individually", "to", "thank", "each", "passenger", ",", "as", "is", "done", "at", "other", "airlines", ",", "Etihad", "for", "example", "]

["The", "staff", "should", "also", "be", "more", "present", "between", "the", "two", "snacks", "to", "meet", "the", "potential", "demands", "of", "passengers", "It", "surprised", "me", "that", "boarding", "nobody", "asks", "me", "to", "present", "my", "visa", ",", "knowing", "that", "in", "case", "of", "refusal", "during", "passport", "control", "the", "return", "is", "at", "the", "expense", "of", "the", "airline", ..."]
Text stemming (different from lemmatization)

```python
import nltk
tokenized_sent = ['The', 'staff', 'should', 'also', 'be', 'more', 'present', 'between', 'the', 'two', 'snacks',
stemmer = nltk.stem.SnowballStemmer('english')
for token in tokenized_sent:
    print(token, " ==> ", stemmer.stem(token))

snacks → snack
potential → potenti
demands → demand
passengers → passing
surprised → surprise
boarding → board
expense → expens
airline → airlin
```
Word distribution – Zipf’s law (Zipfian distribution)

- Zipf’s law (in linguistics):
  
  Given a corpus of natural language utterances, the frequency of any word is inversely proportional to its rank in the frequency table.

- Zipf’s law example (Brown Corpus):
  - Rank #1 word: “the”, accounts for 7% of all word occurrences (69,971 out of about 1 million)
  - Rank #2 word: “of”, account for 3.5% (36,411 occurrences)
  - Rank #3 word: “and” (28,852 occurrences)
  - Only 135 most frequent words accounts for a half of Brown Corpus
Zipf’s law visualization
Stop word and stop word removal

```python
from nltk.corpus import stopwords
print(len(stopwords.words('english')))
print(stopwords.words('english'))

['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', 'you're', 'you've', 'you'll', 'you'd', 'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'his', 'himself', 'she', 'she's', 'her', 'hers', 'herself', 'it', 'it's', 'its', 'itself', 'they', 'them', 'their', 'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', 'that'll', 'these', 'those', 'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having', 'do', 'does', 'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until', 'while', 'of', 'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through', 'during', 'before', 'after', 'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over', 'under', 'again', 'further', 'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both', 'each', 'few', 'more', 'most', 'other', 'some', 'such', 'no', 'nor', 'not', 'only', 'own', 'same', 'so', 'than', 'too', 'very', 's', 't', 'can', 'will', 'just', 'don', 'don't', 'should', 'should've', 'now', 'd', 'll', 'm', 'o', 're', 've', 'y', 'ain', 'aren', 'aren't', 'couldn', 'couldn't', 'didn', 'didn't', 'doesn', 'doesn't', 'hadn', 'hadn't', 'hasn', 'hasn't', 'haven', 'haven't', 'isn', 'isn't', 'ma', 'mightn', 'mightn't', 'mustn', 'mustn't', 'needn', 'needn't', 'shan', 'shan't', 'shouldn', 'shouldn't', 'wasn', 'wasn't', 'weren', 'weren't', 'won', 'won't', 'wouldn', 'wouldn't']
```
## Passenger feedback collection: stemming & stopword removal

### Most frequent words (original data)

<table>
<thead>
<tr>
<th>Word</th>
<th>Freq</th>
<th>Word</th>
<th>Freq</th>
</tr>
</thead>
<tbody>
<tr>
<td>the</td>
<td>1600</td>
<td>flight</td>
<td>285</td>
</tr>
<tr>
<td>to</td>
<td>835</td>
<td>on</td>
<td>278</td>
</tr>
<tr>
<td>and</td>
<td>650</td>
<td>it</td>
<td>219</td>
</tr>
<tr>
<td>i</td>
<td>518</td>
<td>with</td>
<td>206</td>
</tr>
<tr>
<td>of</td>
<td>470</td>
<td>that</td>
<td>203</td>
</tr>
<tr>
<td>a</td>
<td>434</td>
<td>this</td>
<td>189</td>
</tr>
<tr>
<td>in</td>
<td>428</td>
<td>my</td>
<td>181</td>
</tr>
<tr>
<td>was</td>
<td>405</td>
<td>at</td>
<td>179</td>
</tr>
<tr>
<td>not</td>
<td>328</td>
<td>be</td>
<td>174</td>
</tr>
<tr>
<td>for</td>
<td>309</td>
<td>very</td>
<td>169</td>
</tr>
<tr>
<td>is</td>
<td>298</td>
<td>we</td>
<td>166</td>
</tr>
</tbody>
</table>

### Most frequent words (stopwords removed)

<table>
<thead>
<tr>
<th>Word</th>
<th>Freq</th>
<th>Word</th>
<th>Freq</th>
</tr>
</thead>
<tbody>
<tr>
<td>flight</td>
<td>285</td>
<td>time</td>
<td>78</td>
</tr>
<tr>
<td>vietnam</td>
<td>145</td>
<td>passengers</td>
<td>71</td>
</tr>
<tr>
<td>airlines</td>
<td>117</td>
<td>airline</td>
<td>70</td>
</tr>
<tr>
<td>crew</td>
<td>116</td>
<td>seats</td>
<td>69</td>
</tr>
<tr>
<td>staff</td>
<td>110</td>
<td>flights</td>
<td>69</td>
</tr>
<tr>
<td>would</td>
<td>102</td>
<td>economy</td>
<td>66</td>
</tr>
<tr>
<td>service</td>
<td>94</td>
<td>good</td>
<td>66</td>
</tr>
<tr>
<td>cabin</td>
<td>90</td>
<td>food</td>
<td>63</td>
</tr>
<tr>
<td>business</td>
<td>90</td>
<td>airport</td>
<td>60</td>
</tr>
<tr>
<td>class</td>
<td>85</td>
<td>one</td>
<td>59</td>
</tr>
<tr>
<td>seat</td>
<td>82</td>
<td>meal</td>
<td>50</td>
</tr>
</tbody>
</table>

### Most frequent words (stopwords removed, stemming)

<table>
<thead>
<tr>
<th>Word</th>
<th>Freq</th>
<th>Word</th>
<th>Freq</th>
</tr>
</thead>
<tbody>
<tr>
<td>flight</td>
<td>354</td>
<td>busi</td>
<td>90</td>
</tr>
<tr>
<td>airlin</td>
<td>188</td>
<td>class</td>
<td>88</td>
</tr>
<tr>
<td>seat</td>
<td>159</td>
<td>meal</td>
<td>72</td>
</tr>
<tr>
<td>vietnam</td>
<td>145</td>
<td>good</td>
<td>67</td>
</tr>
<tr>
<td>crew</td>
<td>117</td>
<td>economi</td>
<td>66</td>
</tr>
<tr>
<td>staff</td>
<td>113</td>
<td>travel</td>
<td>65</td>
</tr>
<tr>
<td>service</td>
<td>109</td>
<td>one</td>
<td>64</td>
</tr>
<tr>
<td>time</td>
<td>104</td>
<td>food</td>
<td>64</td>
</tr>
<tr>
<td>would</td>
<td>102</td>
<td>airport</td>
<td>62</td>
</tr>
<tr>
<td>passeng</td>
<td>95</td>
<td>check</td>
<td>61</td>
</tr>
<tr>
<td>cabin</td>
<td>91</td>
<td>hour</td>
<td>61</td>
</tr>
</tbody>
</table>
Stop words in aviation domain?

▪ Common English list of stop words
  – About 180 to 200 stop words
  – Domain: general

▪ Stop words in a particular domain
  – Depending on tasks (search, classification, clustering, topic analysis …)

▪ Stop words in aviation domain
  – Also depending on tasks
  – Common stop words include: ‘service’, ‘flight’, ‘Vietnam’, …
Bag-of-word vector representation

- A document is a vector:
  - The length of vector = vocabulary size of the text collection/corpus.
  - Document vectors are very sparse.

- Value of the vector elements?
  - Binary
  - Frequency
  - TF-IDF
Bag-of-word vector representation – binary

Text collection/corpus with three documents, vocabulary size = 18

The elephant sneezed at the sight of potatoes.

Bats can see via echolocation. See the bat sight sneeze!

Wondering, she opened the door to the studio.

1 0 0 0 0 0 1 1 0 1 0 0 1 1 1 0 1 0 0 0
0 1 1 0 1 0 0 0 0 1 0 1 1 0 1 0 1 0 1 0
0 0 0 1 0 0 0 1 0 0 1 0 0 1 1 1 0 1 0 1

at  bat  can  door  echoloc  eleph  of  open  potato  see  she  sight  sneez  studio  the  to  via  wonder
Bag-of-word vector representation – term frequency

Text collection/corpus with three documents, vocabulary size = 18

<table>
<thead>
<tr>
<th></th>
<th>at</th>
<th>bat</th>
<th>can</th>
<th>door</th>
<th>echoloc</th>
<th>eleph</th>
<th>of</th>
<th>open</th>
<th>potato</th>
<th>see</th>
<th>she</th>
<th>sight</th>
<th>sneeze</th>
<th>studio</th>
<th>the</th>
<th>to</th>
<th>via</th>
<th>wonder</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
Bag-of-word vector representation – tf-idf

- Term-frequency – inverse document frequency (tf-idf)
  - Estimating the importance of a word/term in a document within a text collection
  - Used in information retrieval (IR) and text mining
  - Can be used to remove stop-words

- Tf-idf computation
  - Tf: term frequency
  - Idf: inverse document frequency

\[
\text{tfidf}(t, d, D) = \text{tf}(t, d) \times \text{idf}(t, D)
\]

- Where:
  - \(t\) is a particular term
  - \(d\) is a particular document containing term \(t\)
  - \(D\) is the text collection

Different ways to compute \(\text{tf}\) and \(\text{idf}\).
Bag-of-word vector representation – computing tf(t, d)

- Binary:
  \[ tf(t, d) = 1 \text{ if term } t \text{ in document } d, \quad tf(t, d) = 0 \text{ otherwise.} \]

- Raw frequency:
  \[ tf(t, d) = f(t, d) - \text{the frequency of term } t \text{ in } d. \]

- Log normalization (prevent exceptional high-frequency and long documents):
  \[ tf(t, d) = 1 + \log(f(t, d)) \text{ if } t \text{ in } d, \quad tf(t, d) = 0 \text{ otherwise} \]

- Augmented frequency (reduce the influence of long documents):
  \[ tf(t, d) = \alpha + (1 - \alpha) \frac{f(t, d)}{\max_{t' \in d} f(t', d)} \]
Bag-of-word vector representation – computing idf(t, D)

- Log-scaled:

$$idf(t, D) = \log \left( 1 + \frac{|D|}{n_t} \right)$$

- Where

  - $|D|$ is the total number of documents.
  - $n_t$ is the number of documents in $D$ containing term $t$.

- Tf-idf:

$$tfidf(t, d, D) = tf(t, d) \times idf(t, D) = (1 + \log f(t, d)) \times \log \left( 1 + \frac{|D|}{n_t} \right)$$
Bag-of-word vector representation – tf-idf example

\[ \text{tfidf}(t, d, D) = \text{tf}(t, d) \times \text{idf}(t, D) = \left(1 + \log f(t, d)\right) \times \log \left(1 + \frac{|D|}{n_t}\right) \]

<table>
<thead>
<tr>
<th>at</th>
<th>bat</th>
<th>can</th>
<th>door</th>
<th>echoloc</th>
<th>eleph</th>
<th>of</th>
<th>open</th>
<th>potato</th>
<th>see</th>
<th>she</th>
<th>sight</th>
<th>sneez</th>
<th>studio</th>
<th>the</th>
<th>to</th>
<th>via</th>
<th>wonder</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

| 0.96 | 0.0 | 0.0 | 0.0 | 0.0 | 0.96 | 0.96 | 0.0 | 0.96 | 0.0 | 0.0 | 0.64 | 0.64 | 0.0 | 0.76 | 0.0 | 0.0 | 0.0 | 0.0 |
| 0.0 | 1.52 | 0.96 | 0.0 | 0.96 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.64 | 0.64 | 0.0 | 0.48 | 0.0 | 0.96 | 0.0 | 0.0 |
| 0.0 | 0.0 | 0.0 | 0.96 | 0.0 | 0.0 | 0.0 | 0.96 | 0.0 | 0.0 | 0.96 | 0.0 | 0.0 | 0.96 | 0.76 | 0.96 | 0.0 | 0.96 | 0.0 | 0.96 |
N-grams

- 2-grams (bigrams): combinations of two consecutive tokens
- 3-grams (trigrams): combinations of three consecutive tokens
- n-grams: combinations of n consecutive tokens

Bag of n-grams
- Documents can be represented in bag-of-n-grams vectors
- Preserve a certain level of word order information in documents
- N-grams are more complex and richer features than single words
- Remember to eliminate rare n-grams to avoid overfitting
N-grams - example

```python
import nltk

sentence = "The SG and Paris lounge was wonderful, but the Hanoi experience is less than average."
print(sentence)

sentence = nltk.word_tokenize(sentence)
print(sentence)

two_grams = [sentence[i] + '_' + sentence[i + 1] for i in range(len(sentence) - 1)]
print(two_grams)

three_grams = [sentence[i] + '_' + sentence[i + 1] + '_' + sentence[i + 2] for i in range(len(sentence) - 2)]
print(three_grams)
```

Original sentence: The SG and Paris lounge was wonderful, but the Hanoi experience is less than average.

1-grams: ['The', 'SG', 'and', 'Paris', 'lounge', 'was', 'wonderful', ',', 'but', 'the', 'Hanoi', 'experience', 'is', 'less', 'than', 'average', '.']

2-grams: ['The_SG', 'SG_and', 'and_Paris', 'Paris_lounge', 'lounge_was', 'was_wonderful', 'wonderful_', ',', 'but', 'but_the', 'the_Hanoi', 'Hanoi_experience', 'experience_is', 'is_less', 'less_than', 'than_average', 'average_.']

3-grams: ['The_SG_and', 'SG_and_Paris', 'and_Paris_lounge', 'Paris_lounge_was', 'lounge_was_wonderful', 'was_wonderful_', 'wonderful_', 'but', 'but_the', 'but_the_Hanoi', 'the_Hanoi_experience', 'Hanoi_experience_is', 'experience_is_less', 'is_less_than', 'less_than_average', 'than_average_.']
Distributed representations

- **Bag-of-word vector space (frequency and tf-idf)**
  - Vectors are very sparse, leading to keyword mismatch
  - Passenger feedback comments:
    - The vectors are very short and extremely sparse

- **Distributed vector representation**
  - Represent documents (and paragraphs, sentences, words) in a lower-dimensional space
  - The new space is denser, have more topical and semantic information (but black-box)
  - The new space is often more effective (in terms of accuracy)
  - Models and techniques:
    - Word embedding with neural networks (word2vec models: CBOW, skip-gram …)
    - For documents, paragraphs, sentences: doc2vec
    - Latent topic analysis: LSA, pLSA, LDA, NNMF
Training Doc2Vec model on passengers’ feedback texts

- Small corpus of passengers’ feedback texts:
  - Number of feedback comments (documents): **601**
  - The total number of word/term occurrences: **27703** (including punctuations)
  - Number of **uniques** words/tokens: **3417**
  - The total number of sentences: **1546**
  - The average document length (number of words): **46**
  - The average number of sentences per comment: **2.57**
  - The average sentence length: **17.9** (including words and punctuations)

- Doc2Vec model:
  - Number of dimensions: 50
  - Number of training epochs: 100
  - Library: Gensim Doc2Vec
Training Doc2Vec on a corpus of 601 passengers’ comments

from gensim.models.doc2vec import Doc2Vec, TaggedDocument
from nltk.tokenize import word_tokenize

fin = open('data/passenger-feedback-en.txt', mode='r', encoding='utf-8')
corpus = fin.readlines()
fin.close()

tagged_corpus = [TaggedDocument(words=word_tokenize(doc.lower()), tags=[str(idx)]) for idx, doc in enumerate(corpus)]

max_epochs = 100
vec_size = 50
alpha = 0.025

model = Doc2Vec(size=vec_size, alpha=alpha, min_alpha=0.00025, min_count=1, dm=1)
model.build_vocab(tagged_corpus)

for epoch in range(max_epochs):
    print('Iteration {}'.format(epoch))
    model.train(tagged_corpus, total_examples=model.corpus_count, epochs=model.iter)
    model.alpha -= 0.0002
    model.min_alpha = model.alpha

similar_doc = model.docvecs.most_similar(1)
print(similar_doc)

print(model.docvecs[1])
Representation of a comment in Doc2Vec space (dims=50)

**Comment1** = *knowledge of the food served and the wines should be effective.*
Top similar comments in the corpus of a given comment

Comment 1 = knowledge of the food served and the wines should be effective.

- 446: The food was disgusting
- 555: Food needs better. Staff need more fresh.
- 175: The crew should have some knowledge of German.
- 147: Dirty toilets
- 482: Food from melbourne to ho chi min was much better than ho chi min to paris
- 205: Please improve food quality
- 55: Instead of the ordered vegan food, I got an ovo-lacto meal.
- 546: The responsiveness of staff; The ready to serve at anytime with happy face; To listen/willing to talk to customer’s voice; The coordination of ground staff vs. on-board staffs in arranging guest on board.
- 29: Need to speed up procedures.
- 63: The organisation of immigration and security staff at Hanoi is rather chaotic. The shops and restaurants at Hanoi airport are rather poor.
# Summary of text vectorization methods

<table>
<thead>
<tr>
<th>Vectorization Method</th>
<th>Function</th>
<th>Good For</th>
<th>Considerations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frequency</td>
<td>Counts term frequencies</td>
<td>Bayesian models</td>
<td>Most frequent words not always most informative</td>
</tr>
<tr>
<td>One-Hot Encoding</td>
<td>Binarizes term occurrence (0, 1)</td>
<td>Neural networks</td>
<td>All words equidistant, so normalization extra important</td>
</tr>
<tr>
<td>TF–IDF</td>
<td>Normalizes term frequencies across documents</td>
<td>General purpose</td>
<td>Moderately frequent terms may not be representative of document topics</td>
</tr>
<tr>
<td>Distributed Representations</td>
<td>Context-based, continuous term similarity encoding</td>
<td>Modeling more complex relationships</td>
<td>Performance intensive; difficult to scale without additional tools (e.g., Tensorflow)</td>
</tr>
</tbody>
</table>
Part 1: Overview of text data analysis
Part 2: Preprocessing text data and text representation
Part 3: Text classification, clustering, and topic analysis
Part 4: Word and phrase analysis
Part 5: Information extraction from texts
Part 6: Vietnamese text and language processing
Part 7: Text analysis application topics
Part 3: Text classification, clustering, and topic analysis

- Text classification/categorization with
  - Naïve Bayes
  - Support Vector Machines (SVMs)
  - Logistic Regression
  - Classification of feedback comments of airline passengers

- Text clustering with
  - K-means
  - Hierarchical clustering

- Topic analysis with
  - Latent Semantic Analysis/Indexing (LSA/LSI)
  - Latent Dirichlet Allocation (LDA)
  - Non-Negative Matrix Factorization (NNMF)
Part 3: Text classification, clustering, and topic analysis

- Text classification/categorization with
  - Naïve Bayes
  - Support Vector Machines (SVMs)
  - Logistic Regression
  - Classification of feedback comments of airline passengers

- Text clustering with
  - K-means
  - Hierarchical clustering

- Topic analysis with
  - Latent Semantic Analysis/Indexing (LSA/LSI)
  - Latent Dirichlet Allocation (LDA)
  - Non-Negative Matrix Factorization (NNMF)
Text classification workflow

![Text classification workflow diagram](image)
Steps in text classification model construction
Text classification with multinomial Naïve Bayes

- Probability of a document $d$ being in class $c$:

\[
P(c|d) = \frac{P(d|c)P(c)}{P(d)} \propto P(c)P(d|c) = P(c) \prod_{1 \leq k \leq n_d} P(t_k|c)
\]

- The most likely class of document $d$ is:

\[
c_{\text{map}} = \arg\max_{c \in \mathcal{C}} \hat{P}(c|d) = \arg\max_{c \in \mathcal{C}} \hat{P}(c) \prod_{1 \leq k \leq n_d} \hat{P}(t_k|c)
\]

- Using log function:

\[
c_{\text{map}} = \arg\max_{c \in \mathcal{C}} \log \hat{P}(c|d) = \arg\max_{c \in \mathcal{C}} \left[ \log \hat{P}(c) + \sum_{1 \leq k \leq n_d} \log \hat{P}(t_k|c) \right]
\]
```python
from sklearn.datasets import fetch_20newsgroups
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.naive_bayes import MultinomialNB
import numpy as np

# load/download training documents
docs_train = fetch_20newsgroups(subset='train', shuffle=True, random_state=42)
print('Number of training documents: ', len(docs_train.data))

# transform into bag-of-word frequency
freq_vector = CountVectorizer()
X_train_freq = freq_vector.fit_transform(docs_train.data)
print('X_train_freq dimensions: ', X_train_freq.shape)

# transform bag-of-word frequency into bag-of-word tf-idf
tfidf_transformer = TfidfTransformer().fit(X_train_freq)
X_train_tfidf = tfidf_transformer.transform(X_train_freq)
print('X_train_tfidf dimensions: ', X_train_tfidf.shape)

# train a multinomial Naive Bayes classifier
nb_model = MultinomialNB().fit(X_train_tfidf, docs_train.target)
```
# load/download test documents
docs_test = fetch_20newsgroups(subset='test', shuffle=True)

# transform test documents into bag-of-word tf-idf
X_test_freq = freq_vector.transform(docs_test.data)
X_test_tfidf = tfidf_transfomer.transform(X_test_freq)
print('X_test_tfidf dimensions: ', X_test_tfidf.shape)

# predict labels for test documents and compute the accuracy
test_predicted_labels = nb_model.predict(X_test_tfidf)
print('\nTest accuracy = {:.02f}'.format(100 * np.mean(test_predicted_labels == docs_test.target)))

# predict labels for new documents
docs_new = ['God is love.', 'Computer speed is doubled every 18 months.]
X_new_freq = freq_vector.transform(docs_new)
X_new_tfidf = tfidf_transformer.transform(X_new_freq)

new_predicted_labels = nb_model.predict(X_new_tfidf)
for doc, label in zip(docs_new, new_predicted_labels):
    print('{} => {}' % (doc, docs_train.target_names[label]))

Number of training documents: 11314
X_train_freq dimensions: (11314, 130107)
X_train_tfidf dimensions: (11314, 130107)
X_test_tfidf dimensions: (7532, 130107)

Test accuracy = 77.39%

'God is love.' => soc.religion.christian
'Computer speed is doubled every 18 months.' => comp.sys.mac.hardware
from sklearn.datasets import fetch_20newsgroups
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.naive_bayes import MultinomialNB
from sklearn.pipeline import Pipeline
import numpy as np

# load/download training documents
docs_train = fetch_20newsgroups(subset='train', shuffle=True, random_state=42)
print('Number of training documents: ', len(docs_train.data))

# load/download test documents
docs_test = fetch_20newsgroups(subset='test', shuffle=True)

nb_pipeline = Pipeline([
    ('vect', CountVectorizer()),
    ('tfidf', TfidfTransformer()),
    ('nb_model', MultinomialNB()),
])

# preprocess the training data and train the model
nb_pipeline.fit(docs_train.data, docs_train.target)

# predict labels for test documents and compute the accuracy
test_predicted_labels = nb_pipeline.predict(docs_test.data)
print('Test accuracy = ', 100 * np.mean(test_predicted_labels == docs_test.target))
from sklearn.datasets import fetch_20newsgroups
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.linear_model import SGDClassifier
from sklearn.pipeline import Pipeline
from sklearn import import metrics

# load/download training and test datasets
docs_train = fetch_20newsgroups(subset='train', shuffle=True, random_state=42)
docs_test = fetch_20newsgroups(subset='test', shuffle=True)

svm_pipeline = Pipeline([('vect', CountVectorizer()), ('tfidf', TfidfTransformer()), ('svm_model', SGDClassifier(loss='hinge', penalty='l2', alpha=1e-3, random_state=42, max_iter=5, tol=None))])

# preprocess the training data and train the model
svm_pipeline.fit(docs_train.data, docs_train.target)

# predict labels for test documents and compute the accuracy
test_predicted_labels = svm_pipeline.predict(docs_test.data)

# print evaluation report
print(metrics.classification_report(docs_test.target, test_predicted_labels, target_names=docs_test.target_names))
<table>
<thead>
<tr>
<th>category</th>
<th>precision</th>
<th>recall</th>
<th>f1-score</th>
<th>support</th>
</tr>
</thead>
<tbody>
<tr>
<td>alt.atheism</td>
<td>0.73</td>
<td>0.72</td>
<td>0.72</td>
<td>319</td>
</tr>
<tr>
<td>comp.graphics</td>
<td>0.80</td>
<td>0.70</td>
<td>0.74</td>
<td>389</td>
</tr>
<tr>
<td>comp.os.ms-windows.misc</td>
<td>0.73</td>
<td>0.76</td>
<td>0.75</td>
<td>394</td>
</tr>
<tr>
<td>comp.sys.ibm.pc.hardware</td>
<td>0.71</td>
<td>0.70</td>
<td>0.70</td>
<td>392</td>
</tr>
<tr>
<td>comp.sys.mac.hardware</td>
<td>0.83</td>
<td>0.81</td>
<td>0.82</td>
<td>385</td>
</tr>
<tr>
<td>comp.windows.x</td>
<td>0.83</td>
<td>0.77</td>
<td>0.80</td>
<td>395</td>
</tr>
<tr>
<td>misc.forsale</td>
<td>0.84</td>
<td>0.90</td>
<td>0.87</td>
<td>390</td>
</tr>
<tr>
<td>rec.autos</td>
<td>0.92</td>
<td>0.89</td>
<td>0.91</td>
<td>396</td>
</tr>
<tr>
<td>rec.motorcycles</td>
<td>0.92</td>
<td>0.96</td>
<td>0.94</td>
<td>398</td>
</tr>
<tr>
<td>rec.sport.baseball</td>
<td>0.89</td>
<td>0.90</td>
<td>0.89</td>
<td>397</td>
</tr>
<tr>
<td>rec.sport.hockey</td>
<td>0.88</td>
<td>0.99</td>
<td>0.93</td>
<td>399</td>
</tr>
<tr>
<td>sci.crypt</td>
<td>0.83</td>
<td>0.96</td>
<td>0.89</td>
<td>396</td>
</tr>
<tr>
<td>sci.electronics</td>
<td>0.83</td>
<td>0.60</td>
<td>0.70</td>
<td>393</td>
</tr>
<tr>
<td>sci.med</td>
<td>0.87</td>
<td>0.86</td>
<td>0.86</td>
<td>396</td>
</tr>
<tr>
<td>sci.space</td>
<td>0.84</td>
<td>0.96</td>
<td>0.89</td>
<td>394</td>
</tr>
<tr>
<td>soc.religion.christian</td>
<td>0.76</td>
<td>0.94</td>
<td>0.84</td>
<td>398</td>
</tr>
<tr>
<td>talk.politics.guns</td>
<td>0.70</td>
<td>0.92</td>
<td>0.80</td>
<td>364</td>
</tr>
<tr>
<td>talk.politics.mideast</td>
<td>0.90</td>
<td>0.93</td>
<td>0.92</td>
<td>376</td>
</tr>
<tr>
<td>talk.politics.misc</td>
<td>0.89</td>
<td>0.55</td>
<td>0.68</td>
<td>310</td>
</tr>
<tr>
<td>talk.religion.misc</td>
<td>0.85</td>
<td>0.40</td>
<td>0.55</td>
<td>251</td>
</tr>
<tr>
<td>micro avg</td>
<td>0.82</td>
<td>0.82</td>
<td>0.82</td>
<td>7532</td>
</tr>
<tr>
<td>macro avg</td>
<td>0.83</td>
<td>0.81</td>
<td>0.81</td>
<td>7532</td>
</tr>
<tr>
<td>weighted avg</td>
<td>0.83</td>
<td>0.82</td>
<td>0.82</td>
<td>7532</td>
</tr>
</tbody>
</table>

Classification performance of SVM on the 20newsgroups test data
from sklearn.datasets import fetch_20newsgroups
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.pipeline import Pipeline
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import metrics

# load/download training and test datasets
docs_train = fetch_20newsgroups(subset='train', shuffle=True, random_state=42)
docs_test = fetch_20newsgroups(subset='test', shuffle=True)

lr_pipeline = Pipeline([
    ('vect', CountVectorizer()),
    ('tfidf', TfidfTransformer()),
    ('lr_model', LogisticRegression(penalty='l2', random_state=42, max_iter=100)),
])

# preprocess the training data and train the model
lr_pipeline.fit(docs_train.data, docs_train.target)

# predict labels for test documents and compute the accuracy
test_predicted_labels = lr_pipeline.predict(docs_test.data)

# print evaluation report
print(metrics.classification_report(docs_test.target, test_predicted_labels, target_names=docs_test.target_names))
### Classification performance of Logistic Regression on the 20newsgroups test data

<table>
<thead>
<tr>
<th>Category</th>
<th>Precision</th>
<th>Recall</th>
<th>F1-score</th>
<th>Support</th>
</tr>
</thead>
<tbody>
<tr>
<td>alt.atheism</td>
<td>0.80</td>
<td>0.74</td>
<td>0.77</td>
<td>319</td>
</tr>
<tr>
<td>comp.graphics</td>
<td>0.69</td>
<td>0.78</td>
<td>0.74</td>
<td>389</td>
</tr>
<tr>
<td>comp.os.ms-windows.misc</td>
<td>0.76</td>
<td>0.75</td>
<td>0.75</td>
<td>394</td>
</tr>
<tr>
<td>comp.sys.ibm.pc.hardware</td>
<td>0.73</td>
<td>0.72</td>
<td>0.72</td>
<td>392</td>
</tr>
<tr>
<td>comp.sys.mac.hardware</td>
<td>0.81</td>
<td>0.83</td>
<td>0.82</td>
<td>385</td>
</tr>
<tr>
<td>comp.windows.x</td>
<td>0.83</td>
<td>0.74</td>
<td>0.78</td>
<td>395</td>
</tr>
<tr>
<td>misc.forsale</td>
<td>0.76</td>
<td>0.90</td>
<td>0.83</td>
<td>390</td>
</tr>
<tr>
<td>rec.autos</td>
<td>0.91</td>
<td>0.89</td>
<td>0.90</td>
<td>396</td>
</tr>
<tr>
<td>rec.motorcycles</td>
<td>0.94</td>
<td>0.95</td>
<td>0.94</td>
<td>398</td>
</tr>
<tr>
<td>rec.sport.baseball</td>
<td>0.87</td>
<td>0.93</td>
<td>0.90</td>
<td>397</td>
</tr>
<tr>
<td>rec.sport.hockey</td>
<td>0.94</td>
<td>0.96</td>
<td>0.95</td>
<td>399</td>
</tr>
<tr>
<td>sci.crypt</td>
<td>0.93</td>
<td>0.89</td>
<td>0.91</td>
<td>396</td>
</tr>
<tr>
<td>sci.electronics</td>
<td>0.76</td>
<td>0.78</td>
<td>0.77</td>
<td>393</td>
</tr>
<tr>
<td>sci.med</td>
<td>0.89</td>
<td>0.84</td>
<td>0.86</td>
<td>396</td>
</tr>
<tr>
<td>sci.space</td>
<td>0.89</td>
<td>0.92</td>
<td>0.91</td>
<td>394</td>
</tr>
<tr>
<td>soc.religion.christian</td>
<td>0.79</td>
<td>0.93</td>
<td>0.85</td>
<td>398</td>
</tr>
<tr>
<td>talk.politics.guns</td>
<td>0.71</td>
<td>0.90</td>
<td>0.80</td>
<td>364</td>
</tr>
<tr>
<td>talk.politics.mideast</td>
<td>0.96</td>
<td>0.89</td>
<td>0.92</td>
<td>376</td>
</tr>
<tr>
<td>talk.politics.misc</td>
<td>0.79</td>
<td>0.58</td>
<td>0.67</td>
<td>310</td>
</tr>
<tr>
<td>talk.religion.misc</td>
<td>0.83</td>
<td>0.45</td>
<td>0.59</td>
<td>251</td>
</tr>
<tr>
<td><strong>micro avg</strong></td>
<td>0.83</td>
<td>0.83</td>
<td>0.83</td>
<td>7532</td>
</tr>
<tr>
<td><strong>macro avg</strong></td>
<td>0.83</td>
<td>0.82</td>
<td>0.82</td>
<td>7532</td>
</tr>
<tr>
<td><strong>weighted avg</strong></td>
<td>0.83</td>
<td>0.83</td>
<td>0.83</td>
<td>7532</td>
</tr>
</tbody>
</table>
Classification of feedback comments of airline passengers

- **Aspects:**
  - Inflight-service
  - Attendant
  - Food-drink
  - Ground-service
  - Luggage
  - Aircraft-condition
  - Entertainment
  - Language
  - Checkin
  - Online
  - Security
  - Flyer-program
  - Ontime
  - Other

- **Type of sentences:**
  - Suggestion (sugg)
  - Complaint (comp)
  - Recognition (recg)
  - Other (othe)
Annotated feedback comments of airline passengers

- **sugg | attendant | inflight-service** | The flight crew should personalize the contact with the passenger, that is to say when ordering the meal and come individually to thank each passenger, as is done at other airlines (Etihad for example).

- **sugg | attendant | inflight-service** | The staff should also be more present between the two snacks to meet the potential demands of passengers. It surprised me that boarding nobody asks me to present my visa, knowing that in case of refusal during passport control the return is at the expense of the airline ...

- **sugg | food-drink | inflight-service** | knowledge of the food served and the wines should be effective.

- **comp | inflight-service | food-drink** | For being a business flight the on board service was very disappointing, both in terms of service, as well as the choice of food ..... 

- **recg | ground-service | luggage** | My forgotten IPad on the flight was delivered to DANANG - excellent service !!!
Why is classification of feedback comments challenging?

- **sugg | attendant | inflight-service** | The flight crew should personalize the contact with the *passenger*, that is to say when ordering the *meal* and come individually to thank each *passenger*, as is done at other airlines (Etihad for example).

- **sugg | attendant | inflight-service** | The staff should also be more present between the two *snacks* to meet the potential demands of passengers. It surprised me that boarding nobody asks me to present my *visa*, knowing that in case of *refusal* during *passport control* the return is at the expense of the airline ...

- **sugg | food-drink | inflight-service** | knowledge of the food served and the wines should be effective.

- **comp | inflight-service | food-drink** | For being a business flight the on board service was very disappointing, both in terms of service, as well as the choice of food ...

- **recg | ground-service | luggage** | My forgotten IPad on the flight was delivered to DANANG - excellent service !!!
Part 3: Text classification, clustering, and topic analysis

- Text classification/categorization with
  - Naïve Bayes
  - Support Vector Machines (SVMs)
  - Logistic Regression
  - Classification of feedback comments of airline passengers

- Text clustering with
  - K-means
  - Hierarchical clustering

- Topic analysis with
  - Latent Semantic Analysis/Indexing (LSA/LSI)
  - Latent Dirichlet Allocation (LDA)
  - Non-Negative Matrix Factorization (NNMF)
Text clustering workflow
Distance measure

- Let \( d(x, y) \) be a distance measure between data objects \( x \) and \( y \).
- Then \( d(x, y) \) must satisfy four conditions:
  - Non-negativity: \( d(x, y) \geq 0 \)
  - Identity: \( d(x, y) = 0 \) if and only if \( x = y \)
  - Symmetry: \( d(x, y) = d(y, x) \)
  - Triangle inequality: \( d(x, z) \leq d(x, y) + d(y, z) \)
- Some distance functions satisfy all the conditions, some do not.
Euclidean and Manhattan distance

\[ \mathbf{x}_i = (x_{i1}, x_{i2}, \ldots, x_{im}) \]
\[ \mathbf{x}_j = (x_{j1}, x_{j2}, \ldots, x_{jm}) \]

are two data points in \( \mathbb{R}^m \)

\[ d_r(\mathbf{x}_i, \mathbf{x}_j) = \| \mathbf{x}_i - \mathbf{x}_j \|_r = \left( \sum_{k=1}^{m} (|x_{ik} - x_{jk}|)^r \right)^{1/r} \]

Euclidean distance \((r = 2, L_2\text{-norm})\):

\[ d_2(\mathbf{x}_i, \mathbf{x}_j) = \| \mathbf{x}_i - \mathbf{x}_j \|_2 = \sqrt{\sum_{k=1}^{m} (x_{ik} - x_{jk})^2} \]

Manhattan distance \((r = 1, L_1\text{-norm})\):

\[ d_1(\mathbf{x}_i, \mathbf{x}_j) = \| \mathbf{x}_i - \mathbf{x}_j \|_1 = \sum_{k=1}^{m} |x_{ik} - x_{jk}| \]
Cosine similarity

\[
\cos \theta = \frac{x_i^T x_j}{\|x_i\| \cdot \|x_j\|} = \frac{\sum_{k=1}^{m} x_{ik} x_{jk}}{\sqrt{\sum_{k=1}^{m} (x_{ik})^2} \sqrt{\sum_{k=1}^{m} (x_{jk})^2}}
\]

\[
d(x_i, x_j) = 1 - \cos \theta
\]
Jaccard distance

Let $A$ and $B$ be two sets, Jaccard index of $A$ and $B$ is
\[ J(A, B) = \frac{|A \cap B|}{|A \cup B|} \]

Jaccard distance between $A$ and $B$ is
\[ d(A, B) = 1 - J(A, B) = 1 - \frac{|A \cap B|}{|A \cup B|} \]
String edit distance

- Given two strings $s_1$ and $s_2$
- String edit distance between the two strings are the cost to transform one string to the other:
  - Transformation cost is the number of operations:
    - Insert a character (insertion)
    - Delete a character (deletion)
    - Substitute a character by another (substitution)
- Compute string edit distance:
  - Dynamic programming
Hierarchical clustering

- **Agglomerative (bottom-up) hierarchical clustering**
  - Starting: each object/point is a cluster
  - At each iteration: merging the two most related clusters to form a bigger one
  - Stopping criteria:
    - Stop if the number of clusters is equal to what is expected
    - Stop if a resulting cluster is “not good enough”
    - Stop if we reach the final cluster (including all objects)

- **When are two clusters merged?**
  - Centroid-linkage, single-linkages
  - Average-linkage, complete-linkage
  - Radius, diameter
Hierarchical clustering dendrogram
K-means clustering

[nguồn: http://sherrytowers.com/2013/10/24/k-means-clustering]
Clustering passengers’ feedback texts using k-means

- A small set of passengers’ feedback comments: 30 documents
- Number of clusters: 5
- Remove stop words (English standard stop words only)
- Perform stemming
- Document representation: frequency-based
- Dimensionality reduction: No
- Challenges:
  - Passengers’ comments are short and very sparse
The service was handled very quickly and carelessly, as if the flight attendants had very quickly after work. The replenishment of the wine was offered about two minutes after pouring the first glass, about the same time as the coffee. So you can not deal with guests.

Airline personnel appeared extremely disinterested and demotivated throughout, have not seen any of the flight attendants even a smile during the entire 10+ hour flight, nor seen a greeting / farewell of the passengers. The flight attendant directly in front of me did not even consider it necessary to correctly put on the three-point belt, neither at take-off nor landing. I've never experienced such a blatant disregard of security (and ultimately, disrespect for passengers).

The online checkin did not work. I had made several attempts and then gave up annoyed.

Airfrance ticket on this flight because in codeshare. cabin management by the pnc is well below the standard airfrance and therefore disappointing. no passage dissociated drink meal while appetizers are provided. breakfast: the coffee service is done while the passengers have already finished their breakfast without drinks. no presence in the cabin during the flight. no clearing glasses and empty water bottles. no reaction to calls from the seat. Passenger instructions "fasten your belts" always on without justification.

Regarding food service, when one is sitting in row 7 it takes longer for the food to be served. We were disappointed to see (on our connecting flight, not this one) the breakfast breads disappearing to the Premium economy section, so that there was a very limited choice when we were served. Also the bread server decided that we had had enough, when we hadn't been offered a choice - the rest of Business class had, but being the last seat in the section meant we appeared forgotten about. Movies: too many macho films, not enough chick flicks.

There are two issues that I would like to see improved Vietnamairlines: 1 / LHR - HN on 16/02/2017 – Vietnam 54 is a new modern aircraft. But the seats are broken, uncontrollable. 2 / Receptionist HK: do not know how to handle this situation. The attendant's level of professionalism is not satisfactory both in the service of the customer during the flight, the service is confused, not dedicated, ....
Cluster 3

Document 0 ==> travel with an Air France ticket the service on board is below that of air france for the same destination. the cabin crew does not get rid of the glasses and does not make passes during the flight outside of the meals.

Document 1 ==> The crew should definitely improve their English language skills, especially the pronunciation.

Document 6 ==> Sometimes the crew in the airplane did not behave properly that made me very frustrated (They did not help a woman to put up the hand luggage even they were asked)

Document 16 ==> Spanish language in entertainment

Document 21 ==> Air crew professionalism is very poor. Language proficiency need major improvement.

Document 28 ==> Please put some content in Spanish in the plane's entertainment. It is the third language in the world by number of speakers and tourism to Vietnam in Spain is growing
Part 3: Text classification, clustering, and topic analysis

- Text classification/categorization with
  - Naïve Bayes
  - Support Vector Machines (SVMs)
  - Logistic Regression
  - Classification of feedback comments of airline passengers

- Text clustering with
  - Hierarchical clustering
  - K-means

- Topic analysis with
  - Latent Semantic Analysis/Indexing (LSA/LSI)
  - Latent Dirichlet Allocation (LDA)
  - Non-Negative Matrix Factorization (NNMF)
Topic analysis – Latent Semantic Analysis/Indexing (LSA/LSI)

We can represent our corpus with a term-document matrix.

We can then use Singular Value Decomposition (SVD) to factorize into three matrices.
Topic analysis – Latent Dirichlet Allocation (LDA)

The document of a corpus comprise a number of topics.

A topic is a distribution over words.

- aioli fry pancake
- kobe cream garlic
- election stump run
- president Iraqi
- kobe nba record
- draft finals tendon
- note record pop
- cello harmony trio
- fair console game
- unity tactical

A single document invokes multiple topics.
LDA: probabilistic document generation process

(2) Per-document topic distribution generation

(3) Topic sampling for word placeholders

(4) Real word generation
Part 3: Text classification, clustering, and topic analysis

Parameter estimation methods:

- Variational Methods [Blei et al. 2003]
- Expectation-Propagation [Minka & Lafferty 2002]
- Gibbs Sampling [Griffiths & Steyvers 2004]
GibbsLDA++: a C/C++ implementation of LDA

GibbsLDA++ is a C/C++ implementation of Latent Dirichlet Allocation (LDA) using Gibbs Sampling technique for parameter estimation and inference. It is very fast and is designed to analyze hidden latent topic structures of large-scale datasets including large collections of text/web documents. LDA was first introduced by David Blei et al. [Blei03]. There have been several implementations of this model in C (using Variational Methods), Java, and Matlab. We decided to release this implementation of LDA in C/C++ using Gibbs Sampling to provide an alternative to the topic-model community.

GibbsLDA++ is useful for the following potential application areas:

- Information retrieval and search (analyzing semantic/latent topic/concept structures of large text collection for a more intelligent information search)
- Document classification/clustering, document summarization, and text/web mining community in general.
- Content-based image clustering, object recognition, and other applications of computer vision in general.
- Other potential applications in biological data.

Contact us: all comments, suggestions, and bug reports are highly appreciated. And if you have any further problems, please contact us:

Xuan-Hieu Phan (pxhieu at gmail dot com), was at Tohoku University, Japan (now at Vietnam National University, Hanoi).
Cam-Tu Nguyen (incamtu at gmail dot com), was at Vietnam National University, Hanoi (now at Google Japan).

License: GibbsLDA++ is a free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as
<table>
<thead>
<tr>
<th>Topic 0</th>
<th>Topic 1</th>
<th>Topic 2</th>
<th>Topic 10</th>
<th>Topic 7</th>
<th>Topic 18</th>
<th>Topic 19</th>
</tr>
</thead>
<tbody>
<tr>
<td>medical</td>
<td>memory</td>
<td>party</td>
<td>theory</td>
<td>people</td>
<td>economic</td>
<td>film</td>
</tr>
<tr>
<td>health</td>
<td>intel</td>
<td>soviet</td>
<td>quantum</td>
<td>park</td>
<td>trade</td>
<td>films</td>
</tr>
<tr>
<td>medicine</td>
<td>processor</td>
<td>communist</td>
<td>universe</td>
<td>tourism</td>
<td>economy</td>
<td>production</td>
</tr>
<tr>
<td>care</td>
<td>instruction</td>
<td>union</td>
<td>physics</td>
<td>parks</td>
<td>world</td>
<td>movies</td>
</tr>
<tr>
<td>practice</td>
<td>processors</td>
<td>revolution</td>
<td>mechanics</td>
<td>public</td>
<td>government</td>
<td>director</td>
</tr>
<tr>
<td>patient</td>
<td>cpu</td>
<td>war</td>
<td>particles</td>
<td>travel</td>
<td>growth</td>
<td>cinema</td>
</tr>
<tr>
<td>training</td>
<td>performance</td>
<td>political</td>
<td>matter</td>
<td>hotel</td>
<td>countries</td>
<td>studio</td>
</tr>
<tr>
<td>treatment</td>
<td>instructions</td>
<td>government</td>
<td>particle</td>
<td>fair</td>
<td>country</td>
<td>hollywood</td>
</tr>
<tr>
<td>patients</td>
<td>architecture</td>
<td>socialist</td>
<td>relativity</td>
<td>bars</td>
<td>industry</td>
<td>released</td>
</tr>
<tr>
<td>hospital</td>
<td>hardware</td>
<td>power</td>
<td>einstein</td>
<td>activities</td>
<td>foreign</td>
<td>pictures</td>
</tr>
<tr>
<td>surgery</td>
<td>data</td>
<td>stalin</td>
<td>model</td>
<td>bar</td>
<td>production</td>
<td>picture</td>
</tr>
<tr>
<td>clinical</td>
<td>address</td>
<td>revolutionary</td>
<td>space</td>
<td>hotels</td>
<td>sector</td>
<td>studios</td>
</tr>
<tr>
<td>physicians</td>
<td>core</td>
<td>movement</td>
<td>physical</td>
<td>tourist</td>
<td>gdp</td>
<td>directed</td>
</tr>
<tr>
<td>physician</td>
<td>cache</td>
<td>regime</td>
<td>light</td>
<td>visitors</td>
<td>development</td>
<td>motion</td>
</tr>
<tr>
<td>hospitals</td>
<td>computer</td>
<td>parties</td>
<td>classical</td>
<td>home</td>
<td>domestic</td>
<td>release</td>
</tr>
<tr>
<td>doctors</td>
<td>processing</td>
<td>people</td>
<td>field</td>
<td>clubs</td>
<td>billion</td>
<td>sound</td>
</tr>
<tr>
<td>therapy</td>
<td>operating</td>
<td>led</td>
<td>theories</td>
<td>world</td>
<td>industrial</td>
<td>scene</td>
</tr>
<tr>
<td>physical</td>
<td>program</td>
<td>leadership</td>
<td>principle</td>
<td>resort</td>
<td>market</td>
<td>actors</td>
</tr>
<tr>
<td>nursing</td>
<td>registers</td>
<td>republic</td>
<td>energy</td>
<td>outdoor</td>
<td>policy</td>
<td>feature</td>
</tr>
<tr>
<td>doctor</td>
<td>process</td>
<td>lenin</td>
<td>fundamental</td>
<td>restaurants</td>
<td>nations</td>
<td>television</td>
</tr>
<tr>
<td>practitioners</td>
<td>design</td>
<td>communists</td>
<td>experiment</td>
<td>events</td>
<td>international</td>
<td>scenes</td>
</tr>
<tr>
<td>services</td>
<td>mode</td>
<td>world</td>
<td>experiments</td>
<td>pool</td>
<td>industries</td>
<td>industry</td>
</tr>
<tr>
<td>degree</td>
<td>chip</td>
<td>communism</td>
<td>planck</td>
<td>beach</td>
<td>investment</td>
<td>effects</td>
</tr>
<tr>
<td>american</td>
<td>virtual</td>
<td>mao</td>
<td>wave</td>
<td>street</td>
<td>oil</td>
<td>directors</td>
</tr>
<tr>
<td>disease</td>
<td>parallel</td>
<td>military</td>
<td>cosmic</td>
<td>tourists</td>
<td>crisis</td>
<td></td>
</tr>
</tbody>
</table>
### Hidden topic analysis – example in Vietnamese

Sample topics analyzed from VnExpress News Collection. See the complete results online at http://gibbslda.sourceforge.net/vnexpress-200topics.txt

<table>
<thead>
<tr>
<th>Topic 3</th>
<th>Vietnamese</th>
<th>English</th>
</tr>
</thead>
<tbody>
<tr>
<td>bắc sĩ (doctor)</td>
<td>thời trang (fashion)</td>
<td>xác định (identification)</td>
</tr>
<tr>
<td>bệnh viện (hospital)</td>
<td>người mẫu (model)</td>
<td>mẫu (model)</td>
</tr>
<tr>
<td>thuốc (medicine)</td>
<td>mắc (use)</td>
<td>mắc (use)</td>
</tr>
<tr>
<td>bệnh (disease)</td>
<td>trang phục (clothes)</td>
<td>phụ kiện (accessory)</td>
</tr>
<tr>
<td>phẫu thuật (surgery)</td>
<td>thiết kế (design)</td>
<td>thiết kế (design)</td>
</tr>
<tr>
<td>điều trị (treatment)</td>
<td>đẹp (beautiful)</td>
<td>đẹp (beautiful)</td>
</tr>
<tr>
<td>bệnh nhân (patient)</td>
<td>váy (dress)</td>
<td>váy (dress)</td>
</tr>
<tr>
<td>y tế (medical)</td>
<td>đường (collection)</td>
<td>đường (collection)</td>
</tr>
<tr>
<td>ung thư (cancer)</td>
<td>mặc (wear)</td>
<td>mặc (wear)</td>
</tr>
<tr>
<td>tin trong (condition)</td>
<td>mang (wear)</td>
<td>mang (wear)</td>
</tr>
<tr>
<td>cơ thể (body)</td>
<td>phong cách (style)</td>
<td>phong cách (style)</td>
</tr>
<tr>
<td>sức khỏe (health)</td>
<td>quan áo (costume)</td>
<td>quan áo (costume)</td>
</tr>
<tr>
<td>đau (hurt)</td>
<td>nội tiều (famous)</td>
<td>nội tiều (famous)</td>
</tr>
<tr>
<td>gây (cause)</td>
<td>quấn (trousers)</td>
<td>quấn (trousers)</td>
</tr>
<tr>
<td>khám (examine)</td>
<td>trình diễn (perform)</td>
<td>trình diễn (perform)</td>
</tr>
<tr>
<td>kết quả (result)</td>
<td>thích (like)</td>
<td>thích (like)</td>
</tr>
<tr>
<td>cân bệnh (illness)</td>
<td>quyen ru (charming)</td>
<td>quyền lực (powerful)</td>
</tr>
<tr>
<td>nặng (serious)</td>
<td>sang trống (luxurious)</td>
<td>sáng trống (luxurious)</td>
</tr>
<tr>
<td>cho biết (inform)</td>
<td>đẹp (beauty)</td>
<td>đẹp (beauty)</td>
</tr>
<tr>
<td>máu (blood)</td>
<td>gái (girl)</td>
<td>gái (girl)</td>
</tr>
<tr>
<td>xét nghiệm (test)</td>
<td>uông mat (figure)</td>
<td>uông mat (figure)</td>
</tr>
<tr>
<td>chứa (cure)</td>
<td>siêu (super)</td>
<td>siêu (super)</td>
</tr>
<tr>
<td>chứng (trouble)</td>
<td>áo dài (aodai)</td>
<td>áo dài (aodai)</td>
</tr>
<tr>
<td>giầy (shoes)</td>
<td>giày (shoes)</td>
<td>giày (shoes)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Topic 15</th>
<th>Vietnamese</th>
<th>English</th>
</tr>
</thead>
<tbody>
<tr>
<td>thời trang (fashion)</td>
<td>thiết bị (equipment)</td>
<td>thiết bị (equipment)</td>
</tr>
<tr>
<td>người mẫu (model)</td>
<td>sản phẩm (product)</td>
<td>sản phẩm (product)</td>
</tr>
<tr>
<td>mắc (wear)</td>
<td>máy (machine)</td>
<td>máy (machine)</td>
</tr>
<tr>
<td>trang phục (clothes)</td>
<td>màn hình (screen)</td>
<td>màn hình (screen)</td>
</tr>
<tr>
<td>thiết kế (design)</td>
<td>công nghệ (technology)</td>
<td>công nghệ (technology)</td>
</tr>
<tr>
<td>đẹp (beautiful)</td>
<td>điện thoại (telephone)</td>
<td>điện thoại (telephone)</td>
</tr>
<tr>
<td>váy (dress)</td>
<td>hãng (company)</td>
<td>hãng (company)</td>
</tr>
<tr>
<td>đường (collection)</td>
<td>sử dụng (use)</td>
<td>sử dụng (use)</td>
</tr>
<tr>
<td>mặc (wear)</td>
<td>thị trường (market)</td>
<td>thị trường (market)</td>
</tr>
<tr>
<td>mang (wear)</td>
<td>usd (USD)</td>
<td>usd (USD)</td>
</tr>
<tr>
<td>phong cách (style)</td>
<td>pin (battery)</td>
<td>pin (battery)</td>
</tr>
<tr>
<td>quan áo (costume)</td>
<td>cho phép (allow)</td>
<td>cho phép (allow)</td>
</tr>
<tr>
<td>nội tiều (famous)</td>
<td>samsung (Samsung)</td>
<td>samsung (Samsung)</td>
</tr>
<tr>
<td>quấn (trousers)</td>
<td>di động (mobile)</td>
<td>di động (mobile)</td>
</tr>
<tr>
<td>trình diễn (perform)</td>
<td>sony (Sony)</td>
<td>sony (Sony)</td>
</tr>
<tr>
<td>thích (like)</td>
<td>nhạc (music)</td>
<td>nhạc (music)</td>
</tr>
<tr>
<td>quyen ru (charming)</td>
<td>máy tính (computer)</td>
<td>máy tính (computer)</td>
</tr>
<tr>
<td>sang trống (luxurious)</td>
<td>hỗ trợ (support)</td>
<td>hỗ trợ (support)</td>
</tr>
<tr>
<td>đẹp (beauty)</td>
<td>điện tử (electronic)</td>
<td>điện tử (electronic)</td>
</tr>
<tr>
<td>gái (girl)</td>
<td>tính năng (feature)</td>
<td>tính năng (feature)</td>
</tr>
<tr>
<td>uông mat (figure)</td>
<td>kết nối (connect)</td>
<td>kết nối (connect)</td>
</tr>
<tr>
<td>siêu (super)</td>
<td>thiết kế (design)</td>
<td>thiết kế (design)</td>
</tr>
<tr>
<td>áo dài (aodai)</td>
<td>chức năng (function)</td>
<td>chức năng (function)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Topic 44</th>
<th>Vietnamese</th>
<th>English</th>
</tr>
</thead>
<tbody>
<tr>
<td>thiết bị (equipment)</td>
<td>chứng khoán (stock)</td>
<td>chứng khoán (stock)</td>
</tr>
<tr>
<td>sản phẩm (product)</td>
<td>công ty (company)</td>
<td>công ty (company)</td>
</tr>
<tr>
<td>máy (machine)</td>
<td>đầu tư (investment)</td>
<td>đầu tư (investment)</td>
</tr>
<tr>
<td>màn hình (screen)</td>
<td>ngân hàng (bank)</td>
<td>ngân hàng (bank)</td>
</tr>
<tr>
<td>công nghệ (technology)</td>
<td>cổ phần (joint-stock)</td>
<td>cổ phần (joint-stock)</td>
</tr>
<tr>
<td>điện thoại (telephone)</td>
<td>thị trường (market)</td>
<td>thị trường (market)</td>
</tr>
<tr>
<td>hãng (company)</td>
<td>giao dịch (transaction)</td>
<td>giao dịch (transaction)</td>
</tr>
<tr>
<td>sử dụng (use)</td>
<td>dòng (VND)</td>
<td>dòng (VND)</td>
</tr>
<tr>
<td>thị trường (market)</td>
<td>mua (buy)</td>
<td>mua (buy)</td>
</tr>
<tr>
<td>usd (USD)</td>
<td>phát hành (publish)</td>
<td>phát hành (publish)</td>
</tr>
<tr>
<td>pin (battery)</td>
<td>niêm yết (post)</td>
<td>niêm yết (post)</td>
</tr>
<tr>
<td>cho phép (allow)</td>
<td>bán (sell)</td>
<td>bán (sell)</td>
</tr>
<tr>
<td>samsung (Samsung)</td>
<td>tài chính (finance)</td>
<td>tài chính (finance)</td>
</tr>
<tr>
<td>di động (mobile)</td>
<td>đấu giá (auction)</td>
<td>đấu giá (auction)</td>
</tr>
<tr>
<td>sony (Sony)</td>
<td>thông tin (information)</td>
<td>thông tin (information)</td>
</tr>
<tr>
<td>nhạc (music)</td>
<td>doanh nghiệp (business)</td>
<td>doanh nghiệp (business)</td>
</tr>
<tr>
<td>máy tính (computer)</td>
<td>cổ đông (shareholder)</td>
<td>cổ đông (shareholder)</td>
</tr>
<tr>
<td>hỗ trợ (support)</td>
<td>nhà đầu tư (investor)</td>
<td>nhà đầu tư (investor)</td>
</tr>
<tr>
<td>điện tử (electronic)</td>
<td>nhà nước (government)</td>
<td>nhà nước (government)</td>
</tr>
<tr>
<td>tính năng (feature)</td>
<td>tổ chức (organization)</td>
<td>tổ chức (organization)</td>
</tr>
<tr>
<td>kết nối (connect)</td>
<td>triệu (million)</td>
<td>triệu (million)</td>
</tr>
<tr>
<td>thiết kế (design)</td>
<td>quyết (budget)</td>
<td>quyết (budget)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Topic 48</th>
<th>Vietnamese</th>
<th>English</th>
</tr>
</thead>
<tbody>
<tr>
<td>chứng khoán (stock)</td>
<td>bán (sell)</td>
<td>bán (sell)</td>
</tr>
<tr>
<td>công ty (company)</td>
<td>mcdonald (McDonald)</td>
<td>mcdonald (McDonald)</td>
</tr>
<tr>
<td>đầu tư (investment)</td>
<td>thịt (meat)</td>
<td>thịt (meat)</td>
</tr>
<tr>
<td>ngân hàng (bank)</td>
<td>pizza (pizza)</td>
<td>pizza (pizza)</td>
</tr>
<tr>
<td>cổ phần (joint-stock)</td>
<td>bánh mì (bread)</td>
<td>bánh mì (bread)</td>
</tr>
<tr>
<td>thị trường (market)</td>
<td>bánh ngọt (pie)</td>
<td>bánh ngọt (pie)</td>
</tr>
<tr>
<td>giao dịch (transaction)</td>
<td>cửa hàng (shop)</td>
<td>cửa hàng (shop)</td>
</tr>
<tr>
<td>dòng (VND)</td>
<td>thức ăn (food)</td>
<td>thức ăn (food)</td>
</tr>
<tr>
<td>mua (buy)</td>
<td>kem (ice-cream)</td>
<td>kem (ice-cream)</td>
</tr>
<tr>
<td>phát hành (publish)</td>
<td>kem (ice-cream)</td>
<td>kem (ice-cream)</td>
</tr>
<tr>
<td>niêm yết (post)</td>
<td>trà (tea)</td>
<td>trà (tea)</td>
</tr>
<tr>
<td>bán (sell)</td>
<td>mì (noodles)</td>
<td>mì (noodles)</td>
</tr>
<tr>
<td>tài chính (finance)</td>
<td>gà (chicken)</td>
<td>gà (chicken)</td>
</tr>
<tr>
<td>đấu giá (auction)</td>
<td>sô cô la (chocolate)</td>
<td>sô cô la (chocolate)</td>
</tr>
<tr>
<td>thông tin (information)</td>
<td>thức ăn (food)</td>
<td>thức ăn (food)</td>
</tr>
<tr>
<td>doanh nghiệp (business)</td>
<td>thức ăn (food)</td>
<td>thức ăn (food)</td>
</tr>
<tr>
<td>cổ đông (shareholder)</td>
<td>thức ăn (food)</td>
<td>thức ăn (food)</td>
</tr>
<tr>
<td>nhà đầu tư (investor)</td>
<td>thức ăn (food)</td>
<td>thức ăn (food)</td>
</tr>
<tr>
<td>nhà nước (government)</td>
<td>thức ăn (food)</td>
<td>thức ăn (food)</td>
</tr>
<tr>
<td>tổ chức (organization)</td>
<td>thức ăn (food)</td>
<td>thức ăn (food)</td>
</tr>
<tr>
<td>triệu (million)</td>
<td>thức ăn (food)</td>
<td>thức ăn (food)</td>
</tr>
<tr>
<td>quyết (budget)</td>
<td>thức ăn (food)</td>
<td>thức ăn (food)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Topic 56</th>
<th>Vietnamese</th>
<th>English</th>
</tr>
</thead>
<tbody>
<tr>
<td>bán (sell)</td>
<td>thè (card)</td>
<td>thè (card)</td>
</tr>
<tr>
<td>mcdonald (McDonald)</td>
<td>khoa (lock)</td>
<td>khoa (lock)</td>
</tr>
<tr>
<td>thịt (meat)</td>
<td>rút (withdraw)</td>
<td>rút (withdraw)</td>
</tr>
<tr>
<td>pizza (pizza)</td>
<td>chủ (owner)</td>
<td>chủ (owner)</td>
</tr>
<tr>
<td>bánh mì (bread)</td>
<td>chìa (key)</td>
<td>chìa (key)</td>
</tr>
<tr>
<td>bánh ngọt (pie)</td>
<td>thẻ tín dụng (credit card)</td>
<td>thẻ tín dụng (credit card)</td>
</tr>
<tr>
<td>cửa hàng (shop)</td>
<td>atm (ATM)</td>
<td>atm (ATM)</td>
</tr>
<tr>
<td>thức ăn (food)</td>
<td>tín dụng (credit)</td>
<td>tín dụng (credit)</td>
</tr>
<tr>
<td>kem (ice-cream)</td>
<td>thanh toán (pay)</td>
<td>thanh toán (pay)</td>
</tr>
<tr>
<td>trà (tea)</td>
<td>visa (visa)</td>
<td>visa (visa)</td>
</tr>
<tr>
<td>mì (noodles)</td>
<td>tối thiểu (minimum)</td>
<td>tối thiểu (minimum)</td>
</tr>
<tr>
<td>gà (chicken)</td>
<td>mastercard</td>
<td>mastercard</td>
</tr>
<tr>
<td>sô cô la (chocolate)</td>
<td>phát hành (release)</td>
<td>phát hành (release)</td>
</tr>
<tr>
<td>thức ăn (food)</td>
<td>trả nợ (pay debt)</td>
<td>trả nợ (pay debt)</td>
</tr>
<tr>
<td>thức ăn (food)</td>
<td>sả (ready)</td>
<td>sả (ready)</td>
</tr>
<tr>
<td>thức ăn (food)</td>
<td>mật mã (password)</td>
<td>mật mã (password)</td>
</tr>
<tr>
<td>thức ăn (food)</td>
<td>thương niên (annual)</td>
<td>thương niên (annual)</td>
</tr>
<tr>
<td>thức ăn (food)</td>
<td>cành giác (alert)</td>
<td>cành giác (alert)</td>
</tr>
<tr>
<td>thức ăn (food)</td>
<td>chủ thẻ (card owner)</td>
<td>chủ thẻ (card owner)</td>
</tr>
<tr>
<td>thức ăn (food)</td>
<td>theo dõi (follow)</td>
<td>theo dõi (follow)</td>
</tr>
<tr>
<td>thức ăn (food)</td>
<td>nhà băng (bank)</td>
<td>nhà băng (bank)</td>
</tr>
<tr>
<td>thức ăn (food)</td>
<td>tội phạm (criminal)</td>
<td>tội phạm (criminal)</td>
</tr>
<tr>
<td>thức ăn (food)</td>
<td>trộm (steal)</td>
<td>trộm (steal)</td>
</tr>
</tbody>
</table>
Topic analysis – Non-Negative Matrix Factorization (NNMF)

We can represent our corpus with a TF-IDF normalized term-document matrix.

We can then use Non-Negative Matrix Factorization (NNMF) to decompose into two factors whose product approximates the original.
Topic analysis for passengers’ feedback texts with LDA

10 topics, stopword removal, no stemming

**Topic #1** ['class', 'flight', 'food', 'staff', 'good', 'business', 'airlines', 'vietnam']

**Topic #2** ['flight', 'vietnam', 'airport', 'staff', 'airlines', 'airline', 'business', 'check']

**Topic #3** ['flight', 'time', 'crew', 'class', 'economy', 'airlines', 'vietnam', 'check']

**Topic #4** ['flight', 'vietnam', 'business', 'would', 'class', 'airlines', 'sleep', 'staff']

**Topic #5** ['service', 'crew', 'staff', 'vietnam', 'flight', 'thank', 'airlines', 'cabin']

**Topic #6** ['flight', 'ho', 'chi', 'crew', 'cabin', 'seats', 'would', 'minh']

**Topic #7** ['flight', 'meal', 'service', 'crew', 'drink', 'need', 'vn', 'passengers']

**Topic #8** ['flight', 'vietnam', 'flights', 'airlines', 'company', 'would', 'one', 'good']

**Topic #9** ['flight', 'economy', 'premium', 'would', 'seat', 'told', 'service', 'meal']

**Topic #10** ['flight', 'passengers', 'vietnam', 'luggage', 'crew', 'staff', 'airport', 'airlines']
Topic analysis for passengers’ feedback texts with LDA (2)

10 topics, stopword removal, stemming

**Topic #1** ['seat', 'flight', 'good', 'time', 'people', 'crew', 'meal', 'thank']

**Topic #2** ['service', 'flight', 'need', 'class', 'cabin', 'better', 'crew', 'company']

**Topic #3** ['flight', 'seat', 'thank', 'city', 'crew', 'great', 'chi', 'ho']

**Topic #4** ['crew', 'plane', 'service', 'staff', 'passenger', 'flight', 'improvement', 'time']

**Topic #5** ['flight', 'airline', 'vietnam', 'time', 'business', 'check', 'staff', 'would']

**Topic #6** ['flight', 'seat', 'airline', 'vietnam', 'ho', 'chi', 'minh', 'class']

**Topic #7** ['flight', 'airline', 'staff', 'luggage', 'vietnam', 'book', 'hour', 'passenger']

**Topic #8** ['flight', 'service', 'cabin', 'crew', 'air', 'airline', 'class', 'need']

**Topic #9** ['flight', 'food', 'seat', 'cabin', 'drink', 'crew', 'armrest', 'control']

**Topic #10** ['airline', 'vietnam', 'service', 'meal', 'staff', 'flight', 'company', 'best']
Topic analysis for passengers’ feedback texts with LDA (3)

10 topics, stopword removal, no stemming, domain stop word removal

Topic #1 ['class', 'business', 'emergency', 'minutes', 'aircraft', 'passengers', 'meal', 'melbourne']
Topic #2 ['drink', 'class', 'meal', 'travel', 'lounge', 'wine', 'international', 'skills']
Topic #3 ['good', 'business', 'class', 'check', 'fly', 'experience', 'work', 'dreamliner']
Topic #4 ['thank', 'frequent', 'class', 'flyer', 'armrest', 'meal', 'entertainment', 'control']
Topic #5 ['program', 'passengers', 'english', 'row', 'business', 'trip', 'offered', 'water']
Topic #6 ['economy', 'class', 'business', 'premium', 'luggage', 'good', 'online', 'passengers']
Topic #7 ['food', 'economy', 'business', 'class', 'check', 'good', 'journey', 'lounge']
Topic #8 ['board', 'passengers', 'luggage', 'thank', 'card', 'friendly', 'food', 'world']
Topic #9 ['business', 'class', 'food', 'economy', 'passengers', 'good', 'premium', 'vna']
Topic #10 ['passengers', 'meal', 'food', 'check', 'trip', 'good', 'lounge', 'in']
Topic analysis for passengers’ feedback texts with LDA (4)

10 topics, stopword removal, stemming, domain stop word removal

**Topic #1** ['travel', 'busi', 'fli', 'check', 'class', 'thank', 'help', 'passeng']

**Topic #2** ['improv', 'passeng', 'luggag', 'connect', 'qualiti', 'departur', 'late', 'check']

**Topic #3** ['work', 'entertain', 'system', 'program', 'book', 'board', 'tri', 'loung']

**Topic #4** ['food', 'meal', 'choic', 'experi', 'entertain', 'languag', 'busi', 'good']

**Topic #5** ['luggag', 'passeng', 'understand', 'check', 'custom', 'improv', 'book', 'french']

**Topic #6** ['class', 'busi', 'econom', 'member', 'meal', 'premium', 'prioriti', 'travel']

**Topic #7** ['good', 'sleep', 'offer', 'food', 'drink', 'econom', 'fli', 'meal']

**Topic #8** ['point', 'passeng', 'friend', 'board', 'food', 'check', 'meal', 'toilet']

**Topic #9** ['book', 'econom', 'passeng', 'row', 'premium', 'check', 'request', 'board']

**Topic #10** ['meal', 'serv', 'ticket', 'travel', 'check', 'food', 'passeng', 'loung']
15 topics, stopword removal, no stemming, domain stop word removal

Topic #1 ['friendly', 'smile', 'drinks', 'business', 'entertainment', 'vna', 'night', 'row']
Topic #2 ['meal', 'drink', 'food', 'well', 'served', 'passengers', 'luggage', 'quality']
Topic #3 ['thank', 'good', 'trip', 'business', 'perfect', 'available', 'flyer', 'frequent']
Topic #4 ['passengers', 'trip', 'sleep', 'good', 'entertainment', 'meal', 'wine', 'bag']
Topic #5 ['food', 'good', 'language', 'quality', 'improve', 'meal', 'breakfast', 'served']
Topic #6 ['food', 'good', 'in', 'toilet', 'check', 'online', 'available', 'quality']
Topic #7 ['thanks', 'book', 'around', 'online', 'water', 'booking', 'tried', 'check']
Topic #8 ['economy', 'luggage', 'broken', 'may', 'friendly', 'fly', 'help', 'baggage']
Topic #9 ['business', 'luggage', 'fly', 'thank', 'check', 'priority', 'lounge', 'passengers']
Topic #10 ['work', 'board', 'good', 'luggage', 'entertainment', 'connecting', 'pleasant', 'food']
Topic #11 ['passengers', 'children', 'boarding', 'water', 'thank', 'leg', 'sleep', 'economy']
Topic #12 ['passengers', 'check', 'food', 'helpful', 'snacks', 'business', 'meals', 'haul']
Topic #13 ['check', 'economy', 'business', 'french', 'choice', 'enough', 'class', 'breakfast']
Topic #14 ['class', 'business', 'economy', 'premium', 'aircraft', 'passengers', 'fly', 'row']
Topic #15 ['good', 'economy', 'experience', 'class', 'premium', 'armrest', 'business', 'sleep']
15 topics, stopword removal, stemming, domain stop word removal

Topic #1 ['custom', 'busi', 'class', 'sleep', 'fli', 'delay', 'good', 'member']
Topic #2 ['meal', 'drink', 'serv', 'offer', 'wine', 'class', 'busi', 'economi']
Topic #3 ['passeng', 'busi', 'attend', 'class', 'atten', 'departur', 'food', 'prioriti']
Topic #4 ['book', 'good', 'tri', 'improv', 'passeng', 'custom', 'economi', 'luggag']
Topic #5 ['economi', 'food', 'class', 'premium', 'min', 'help', 'london', 'good']
Topic #6 ['languag', 'pleasant', 'upgrad', 'spanish', 'entertain', 'world', 'domest', 'surpris']
Topic #7 ['fli', 'travel', 'busi', 'year', 'frequent', 'loung', 'impress', 'trip']
Topic #8 ['check', 'passeng', 'profession', 'luggag', 'good', 'travel', 'intern', 'aircraft']
Topic #9 ['class', 'meal', 'economi', 'french', 'smile', 'premium', 'suitcas', 'correct']
Topic #10 ['breakfast', 'travel', 'serv', 'food', 'offer', 'economi', 'feel', 'check']
Topic #11 ['food', 'good', 'qualiti', 'improv', 'trip', 'meal', 'board', 'high']
Topic #12 ['thank', 'friend', 'fli', 'board', 'help', 'meal', 'good', 'book']
Topic #13 ['passeng', 'toilet', 'check', 'luggag', 'drink', 'vna', 'inform', 'busi']
Topic #14 ['entertain', 'work', 'loung', 'system', 'program', 'improv', 'movi', 'skill']
Topic #15 ['check', 'class', 'economi', 'busi', 'passeng', 'meal', 'board', 'experi']
Analyzing passengers’ comments: challenges and solutions

▪ Short and sparse data
  – Comments are very short
  – Bag of word vectors are very sparse
  – Very low co-occurrence

▪ How to improve classification and clustering performance
  – Using external data (using a larger external collection of aviation-related texts)
  – Expanding passengers’ comments using semantic lexical databases like WordNet
    ▪ Get more synonyms
    ▪ Get hypernyms and hyponyms
Contents

- Part 1: Overview of text data analysis
- Part 2: Preprocessing text data and text representation
- Part 3: Text classification, clustering, and topic analysis
- Part 4: Word and phrase analysis
- Part 5: Information extraction from texts
- Part 6: Vietnamese text and language processing
- Part 7: Text analysis application topics
Part 4: Word and phrase analysis

- WordNet
  - Senses and synonyms
  - The WordNet hierarchy and lexical relations
  - Semantic similarity

- Collocations
  - Finding significant collocations
  - Using collocations as rich and complex features
What is WordNet?

- WordNet is a semantically oriented dictionary of English
  - In other words, a lexical database for English
  - Has hierarchical structure
  - Nouns, verbs, adjectives, and adverbs are grouped into synonym sets (synsets)
  - Each synset expresses a distinct concept
  - Synsets are inter-linked by means of conceptual-semantic and lexical relations

- NLTK implement English WordNet, includes
  - 155,287 words
  - 117,659 synonym sets
WordNet senses and synonyms

```python
>>> from nltk.corpus import wordnet as wn

>>> wn.synsets('attendant')
[Synset('attendant.n.01'), Synset('attendant.n.02'), Synset('accompaniment.n.01'), ...]

>>> wn.synset('attendant.n.01').definition()
'someone who waits on or tends to or attends to the needs of another'

>>> wn.synset('attendant.n.02').definition()
'a person who is present and participates in a meeting'

>>> wn.synset('attendant.n.01').lemmas()
[Lemma('attendant.n.01.attendant'), Lemma('attendant.n.01.attender'), Lemma('attendant.n.01.tender')]
```
WordNet hierarchical structure

- **Hypernym**
  - $X$ is a hypernym of $Y$ means: $Y$ is a type of $X$
  - ‘attendant’ is a type of ‘assistant’ (according to WordNet)

- **Hyponym**
  - $X$ is a hyponym of $Y$ means: $X$ is a type of $Y$
  - ‘secretary’ is a type of ‘assistant’

- **Hypernym and hyponym are two types of lexical relations in WordNet**
  - Connecting synsets
  - Can go up and go down from a synset by following hypernym or hyponym relations
>>> attendant = wn.synset('attendant.n.01')

>>> attendant.hypernyms()
[Synset('assistant.n.01')]

>>> assistant = wn.synset('assistant.n.01')

>>> assistant.hyponyms()
[Synset('accomplice.n.01'), Synset('aide.n.02'), Synset('attendant.n.01'), Synset('bat_boy.n.01'), Synset('coadjutor.n.01'), Synset('dental_assistant.n.01'), Synset('deputy.n.02'), Synset('dresser.n.03'), Synset('event_planner.n.01'), Synset('facilitator.n.01'), Synset('flower_girl.n.02'), Synset('girl_friday.n.01'), Synset('hatchet_man.n.02'), Synset('instrument.n.03'), Synset('labor_coach.n.01'), Synset('mannequin.n.01'), Synset('model.n.03'), Synset('paraprofessional.n.01'), Synset('powder_monkey.n.01'), Synset('prompter.n.01'), Synset('right-hand_man.n.01'), Synset('secretary.n.02'), Synset('sidesman.n.01'), Synset('subordinate.n.01'), Synset('underboss.n.01'), Synset('water_boy.n.01'), Synset('whipper-in.n.01')]

>>> attendant.hyponyms()
[Synset('baggageman.n.01'), Synset('batman.n.01'), Synset('bellboy.n.01'), Synset('bridesmaid.n.01'), Synset('caddie.n.01'), Synset('checker.n.01'), Synset('companion.n.03'), Synset('courtier.n.01'), Synset('cupbearer.n.01'), Synset('equerry.n.02'), Synset('escort.n.03'), Synset('esquire.n.01'), Synset('famulus.n.01'), Synset('gillie.n.01'), Synset('groomsman.n.01'), Synset('lifeguard.n.01'), Synset('linkboy.n.01'), Synset('loader.n.02'), Synset('matron_of_honor.n.01'), Synset('orderly.n.01'), Synset('orderly.n.02'), Synset('page.n.05'), Synset('page.n.06'), Synset('racker.n.01'), Synset('rocker.n.01'), Synset('second.n.07'), Synset('squire.n.01'), Synset('squire.n.03'), Synset('steward.n.03'), Synset('stretcher-bearer.n.01'), Synset('trainbearer.n.01'), Synset('waker.n.01')]
Semantic similarity

- Knowing which words are semantically related is useful
  - Measuring the semantic similarity between words, sentences, or documents
  - Useful for indexing, classification, clustering of (short and sparse) documents

```python
>>> coffee = wn.synset('coffee.n.01')
>>> wine = wn.synset('wine.n.01')
>>> beer = wn.synset('beer.n.01')
>>> juice = wn.synset('juice.n.01')
>>> luggage = wn.synset('luggage.n.01')

>>> coffee.path_similarity(wine)
0.25
>>> coffee.path_similarity(beer)
0.2
>>> coffee.path_similarity(juice)
0.2
>>> wine.path_similarity(beer)
0.25
>>> wine.path_similarity(juice)
0.167
>>> beer.path_similarity(juice)
0.143
>>> coffee.path_similarity(luggage)
0.078
```
Collocations

- **N-grams and using n-grams**
  - N-grams are normally combinations of consecutive words/tokens
  - Not all of n-grams are useful
  - Using n-grams directly causes overfitting

- **Significant collocations:**
  - Collocations are sequences of consecutive words/tokens whose likelihood of co-occurrence is caused by something other than random chance.

- **Measure of likelihood of co-occurrence or association:**
  - PMI: pointwise mutual information
  - Hypothesis testing: e.g., chi-squared test
import nltk
from nltk.collocations import *
from nltk import sent_tokenize, word_tokenize

fin = open('data/passenger-feedback-en.txt', mode='r', encoding='utf-8')
corpus = fin.readlines()
fin.close()

sentences = [sent for doc in corpus for sent in sent_tokenize(doc)]
trigrams = [''.join(sentence) for sentence in sentences]
words = ' '.join(sentences).lower().split()

trigram_measures = nltk.collocations.TrigramAssocMeasures()
trigram_finder = TrigramCollocationFinder.from_words(words)

trigram_finder.apply_freq_filter(3)
colls = trigram_finder.score_ngrams(trigram_measures.chi_sq)

for ngram, score in colls:
    print('{}	{}'.format(repr(ngram), score))
Top 30 bi-gram collocations (using chi-squared test)

('da', 'nang') 27703.0 ('emergency', 'exit') 8902.071135900193
('ho', 'chi') 26381.902007666711 ('minh', 'city') 8428.791220573616
('siem', 'reap') 23744.57124288604 ('&', '#') 7691.6660147944895
('chi', 'minh') 21778.031438531078 ('pay', 'attention') 7665.9679489316895
('noi', 'bai') 20776.499918772563 ('cabin', 'crew') 7403.131230764835
('#', 'thirteen') 16620.599870036098 ('boeing', '787') 6922.999684130517
('call', 'button') 16620.599870036098 ('option', 'town') 6390.692057761733
('frequent', 'flyer') 15918.629733772143 ('airbus', 'a350') 6328.456583816831
('pleasantly', 'surprised') 15388.332932181225 ('$', '350') 5933.142540030518
('vietnam', 'airlines') 12019.318530402581 ('thank', 'you') 5883.812274563142
('formal', 'complaint') 11870.999814337289 ('sky', 'team') 5537.399682372007
('flying', 'blue') 11714.151325292503 ('rather', 'than') 5309.67817253104
('premium', 'economy') 10722.491947697408 ('checked', 'baggage') 5241.135540525463
('business', 'class') 10518.347519094465 ('into', 'account') 5191.312175125458
('air', 'france') 10245.91487006305 ('thirteen', '&') 4614.666395908544
Top 30 tri-gram collocations (using pmi)

<table>
<thead>
<tr>
<th>Collocation</th>
<th>PMI</th>
</tr>
</thead>
<tbody>
<tr>
<td>('#', 'thirteen', '&amp;')</td>
<td>23.0236</td>
</tr>
<tr>
<td>('&amp;', '#', 'thirteen')</td>
<td>23.0236</td>
</tr>
<tr>
<td>('coffee', 'or', 'tea')</td>
<td>19.9841</td>
</tr>
<tr>
<td>('chi', 'min', 'city')</td>
<td>18.7308</td>
</tr>
<tr>
<td>('ho', 'chi', 'minh')</td>
<td>18.7163</td>
</tr>
<tr>
<td>('24', 'hours', 'before')</td>
<td>18.4508</td>
</tr>
<tr>
<td>('chi', 'minh', 'city')</td>
<td>18.4089</td>
</tr>
<tr>
<td>('ho', 'chi', 'min')</td>
<td>18.3158</td>
</tr>
<tr>
<td>('value', 'for', 'money')</td>
<td>17.0216</td>
</tr>
<tr>
<td>('long', 'haul', 'flights')</td>
<td>16.8270</td>
</tr>
<tr>
<td>('small', 'bottle', 'of')</td>
<td>16.0540</td>
</tr>
<tr>
<td>('can', 'be', 'adjusted')</td>
<td>15.9093</td>
</tr>
<tr>
<td>('for', 'smaller', 'people')</td>
<td>15.8864</td>
</tr>
<tr>
<td>('pick', 'up', 'my')</td>
<td>15.7551</td>
</tr>
<tr>
<td>('would', 'have', 'liked')</td>
<td>15.4409</td>
</tr>
<tr>
<td>('much', 'better', 'than')</td>
<td>15.4112</td>
</tr>
<tr>
<td>('pay', 'attention', 'to')</td>
<td>15.3724</td>
</tr>
<tr>
<td>('the', 'boeing', '787')</td>
<td>15.2866</td>
</tr>
<tr>
<td>('bottle', 'of', 'water')</td>
<td>15.2239</td>
</tr>
<tr>
<td>('12', 'hour', 'flight')</td>
<td>15.1127</td>
</tr>
<tr>
<td>('run', 'out', 'of')</td>
<td>15.0578</td>
</tr>
<tr>
<td>('as', 'soon', 'as')</td>
<td>14.8541</td>
</tr>
<tr>
<td>('not', 'recognised', 'by')</td>
<td>14.6719</td>
</tr>
<tr>
<td>('chi', 'minh', 'airport')</td>
<td>14.5060</td>
</tr>
<tr>
<td>('a', '12', 'hour')</td>
<td>14.5060</td>
</tr>
<tr>
<td>('times', 'a', 'year')</td>
<td>14.4685</td>
</tr>
<tr>
<td>('fly', 'with', 'va')</td>
<td>14.4196</td>
</tr>
<tr>
<td>('keep', 'it', 'up')</td>
<td>14.3426</td>
</tr>
<tr>
<td>('the', 'airbus', 'a350')</td>
<td>14.3273</td>
</tr>
<tr>
<td>('from', 'ho', 'chi')</td>
<td>14.2163</td>
</tr>
</tbody>
</table>
Top 30 quad-gram collocations (using chi-squared test)

('&', '#', 'thirteen', '&') 39371926051.77
('ho', 'chi', 'minh', 'city') 3860788781.191
('ho', 'chi', 'min', 'city') 683210577.0617
('in', 'ho', 'chi', 'minh') 76067284.67535
('ho', 'chi', 'minh', 'airport') 66009971.48212
('to', 'ho', 'chi', 'minh') 58912611.95206
('from', 'ho', 'chi', 'minh') 45691059.83183
('needs', 'to', 'be', 'improved') 17224691.12499
('24', 'hours', 'before', 'the') 14054637.37101
('chi', 'minh', 'city', 'and') 12117384.66473
('would', 'be', 'nice', 'if') 10919862.70882
('i', 'would', 'have', 'liked') 8526216.596513
('it', 'would', 'have', 'liked') 7628414.647684
('a', '12', 'hour', 'flight') 6785204.933756
('was', 'not', 'recognised', 'by') 6078015.019319
('flying', 'with', 'vietnam', 'airlines') 5954976.99717
('to', 'pick', 'up', 'my') 5504785.92198
('would', 'have', 'liked', 'to') 3432545.56593
('on', 'vietnam', 'airlines', 'website') 2269582.61398
('to', 'fly', 'with', 'va') 2182525.72813
('it', 'would', 'be', 'great') 1411926.99827
('pay', 'attention', 'to', 'the') 1169270.92241
('the', 'premium', 'economy', 'class') 962970.112421
('we', 'had', 'to', 'wait') 836878.714880
('as', 'well', 'as', 'the') 823402.371284
('hours', 'before', 'the', 'flight') 817185.254893
('for', 'a', 'long', 'time') 798281.270698
('satisfied', 'with', 'the', 'service') 772384.182663
('at', 'the', 'same', 'time') 743491.619654
('would', 'be', 'nice', 'to') 700767.519331
Collocations as features for classification and clustering

- Reduce number of n-grams
- Rich and meaningful features
- Avoid overfitting
- Collocations are also useful for
  - Information extraction (e.g., named entity recognition)
Contents

- **Part 1**: Overview of text data analysis
- **Part 2**: Preprocessing text data and text representation
- **Part 3**: Text classification, clustering, and topic analysis
- **Part 4**: Word and phrase analysis
- **Part 5**: Information extraction from texts
- **Part 6**: Vietnamese text and language processing
- **Part 7**: Text analysis application topics
Part 5: Information extraction (IE) from texts

- Why information extraction from texts?
- Information extraction pipeline
- Sentence segmentation
- Word tokenization
- Part-of-speech tagging
- Named entity recognition
  - Named entity types in aviation
- Relation extraction
Why information extraction from texts?

- Smart search
  - Job search
  - Product search, e.g., websosan
- Knowledge base/graph
- Question answering
- Semantic web
Examples of IE

William (Bill) H. Gates, born on October 28, 1955 and grew up in Seattle, Washington, is the co-founder, chairman, and chief software architect of Microsoft Corporation, the worldwide leader in software, services, and solutions that help people and businesses realize their full potential. Microsoft had revenues of US$39.79 billion for the fiscal year ending June, 2005, and employs more than 61,000 people in 102 countries and regions. According to Forbes, Bill Gates is the richest person in the world with a net worth of about US$50 billion, as of March, 2006.
IE and different levels of NLP and NLU

A dog is chasing a boy on the playground

- String of characters
- Sequence of words + POS tags
- + Syntactic structures
- + Entities and relations
- + Logic predicates
- + Speech acts

Deeper NLP: requires more human effort; less accurate

Closer to knowledge representation
A common pipeline for information extraction from texts
Example of sentence segmentation

document = "Everything was wonderful. The crew was fantastic. I felt I got special treatment and individual attention. The long flight was a wonderful experience. I would only suggest that the movie selection is changed more often, as I got the same selection on the return flight. The second is the business lounge in Hanoi, the only negative point of this trip. The SG and Paris lounge was wonderful, but the Hanoi experience is less than average. Otherwise, it is money for buck and a wonderful flight. I am considering becoming a Vietnam Airlines frequent flyer."

import nltk
document = "Everything was wonderful. The crew was fantastic. I felt I got special treatment and individual attention. The long flight was a wonderful experience. I would only suggest that the movie selection is changed more often, as I got the same selection on the return flight. The second is the business lounge in Hanoi, the only negative point of this trip. The SG and Paris lounge was wonderful, but the Hanoi experience is less than average. Otherwise, it is money for buck and a wonderful flight. I am considering becoming a Vietnam Airlines frequent flyer."
Example of sentence segmentation (result)

1. Everything was wonderful.
2. The crew was fantastic.
3. I felt I got special treatment and individual attention.
4. The long flight was a wonderful experience.
5. I would only suggest that the movie selection is changed more often, as I got the same selection on the return flight.
6. The second is the business loung in Hanoi, the only negative point of this trip.
7. The SG and Paris lounge was wonderful, but the Hanoi experience is less than average.
8. Otherwise, it is money for buck and a wonderful flight.
9. I am considering becoming a Vietnam Airlines frequent flyer.
Example of word tokenization

```python
print()

sentences = [nltk.word_tokenize(sentence) for sentence in sentences]
for sentence in sentences:
    print(' '.join([word for word in sentence]))
```

- Everything was wonderful.
- The crew was fantastic.
- I felt I got special treatment and individual attention.
- The long flight was a wonderful experience.
- I would only suggest that the movie selection is changed more often, as I got the same selection on the return flight.
- The second is the business lounge in Hanoi, the only negative point of this trip.
- The SG and Paris lounge was wonderful, but the Hanoi experience is less than average.
- Otherwise, it is money for buck and a wonderful flight.
- I am considering becoming a Vietnam Airlines frequent flyer.
Example of part-of-speech tagging (POS tagging)

```python
print()
sentences = [nltk.pos_tag(sentence) for sentence in sentences]
for sentence in sentences:
    print(' '.join([word + '/' + tag for (word, tag) in sentence]))
```

- Everything/NN was/VBD wonderful/JJ ./
- The/DT crew/NN was/VBD fantastic/JJ ./
- I/PRP felt/VBD I/PRP got/VBD special/JJ treatment/NN and/CC individual/JJ attention/NN ./
- The/DT long/JJ flight/NN was/VBD a/DT wonderful/JJ experience/NN ./
- I/PRP would/MD only/RB suggest/VB that/IN the/DT movie/NN selection/NN is/VBZ changed/VBN more/RBR often/RB ./, as/IN I/PRP got/VBD the/DT same/JJ selection/NN on/IN the/DT return/NN flight/NN ./
- The/DT second/JJ is/VBZ the/DT business/NN loung/NN in/IN Hanoi/NNP ./, the/DT only/JJ negative/JJ point/NN of/IN this/DT trip/NN ./
- The/DT SG/NNP and/CC Paris/NNP lounge/NN was/VBD wonderful/JJ ./, but/CC the/DT Hanoi/NNP experience/NN is/VBZ less/JJR than/IN average/NN ./
- Otherwise/RB ./, it/PRP is/VBZ money/NN for/IN buck/NN and/CC a/DT wonderful/JJ flight/NN ./
- I/PRP am/VBP considering/VBG becoming/VBG a/DT Vietnam/NNP Airlines/NNPS frequent/JJ flyer/NN ./
Named entity recognition (NER)

- Using NLTK pre-trained NER
  - PERSON
  - ORGANIZATION
  - GPE (geopolitical entity)
The second lounge was wonderful, but the experience is less than average.

I am considering becoming a frequent flyer.

The business lounge in Hanoi is the only negative point of this trip.
Basic named entity types

- Seven types of named entity in MUC
  - ORGANIZATION
  - PERSON
  - LOCATION
  - DATE/TIME
  - NUMBER
  - MONEY
  - PERCENT
# Named entity types in spaCy

<table>
<thead>
<tr>
<th>NE type</th>
<th>Description</th>
<th>NE type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PERSON</td>
<td>People, including fictional.</td>
<td>LAW</td>
<td>Named documents made into laws.</td>
</tr>
<tr>
<td>NORP</td>
<td>Nationalities or religious or political groups.</td>
<td>LANGUAGE</td>
<td>Any named language.</td>
</tr>
<tr>
<td>FAC</td>
<td>Buildings, airports, highways, bridges, etc.</td>
<td>DATE</td>
<td>Absolute or relative dates or periods.</td>
</tr>
<tr>
<td>ORG</td>
<td>Companies, agencies, institutions, etc.</td>
<td>TIME</td>
<td>Times smaller than a day.</td>
</tr>
<tr>
<td>GPE</td>
<td>Countries, cities, states.</td>
<td>PERCENT</td>
<td>Percentage, including &quot;%&quot;.</td>
</tr>
<tr>
<td>LOC</td>
<td>Non-GPE locations, mountain ranges, bodies of water.</td>
<td>MONEY</td>
<td>Monetary values, including unit.</td>
</tr>
<tr>
<td>PRODUCT</td>
<td>Objects, vehicles, foods, etc. (Not services.)</td>
<td>QUANTITY</td>
<td>Measurements, as of weight or distance.</td>
</tr>
<tr>
<td>EVENT</td>
<td>Named hurricanes, battles, wars, sports events, etc.</td>
<td>ORDINAL</td>
<td>&quot;first&quot;, &quot;second&quot;, etc.</td>
</tr>
<tr>
<td>WORK_OF_ART</td>
<td>Titles of books, songs, etc.</td>
<td>CARDINAL</td>
<td>Numerals that do not fall under another type.</td>
</tr>
</tbody>
</table>
# Named entity types in aviation

<table>
<thead>
<tr>
<th>NE type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PERSON</td>
<td>Names of passengers, cabin crew members ...</td>
</tr>
<tr>
<td>DATE/TIME</td>
<td>Date and time</td>
</tr>
<tr>
<td>CITY</td>
<td>City name, e.g., Ho Chi Minh city, New York ...</td>
</tr>
<tr>
<td>AIRPORT</td>
<td>Names of airports, e.g., Tan Son Nhat, Noi Bai, Narita ...</td>
</tr>
<tr>
<td>AIRLINE</td>
<td>e.g., Vietnam Airlines, JAL, Korean Air ...</td>
</tr>
<tr>
<td>FLIGHT-ROUTE</td>
<td>Flying routes, e.g., Hanoi-Bangkok, Paris-Siem Reap ...</td>
</tr>
<tr>
<td>FLYER-PROGRAM</td>
<td>e.g., Platinum, Gold, Sliver ...</td>
</tr>
<tr>
<td>AIRCRAFT</td>
<td>e.g., Airbus A321-200, Boeing 787-9 ...</td>
</tr>
<tr>
<td>FLIGHT</td>
<td>e.g., VN773, JL 751, VN-209 ...</td>
</tr>
<tr>
<td>TICKET-CLASS</td>
<td>e.g., business, economy ...</td>
</tr>
<tr>
<td>FOOD</td>
<td>Types of food served in cabin</td>
</tr>
<tr>
<td>DRINK</td>
<td>Type of drink served in cabin</td>
</tr>
</tbody>
</table>
Coreference resolution and information extraction

William (Bill) H. Gates, born on October 28, 1955 and grew up in Seattle, Washington, is the co-founder, chairman, and chief software architect of Microsoft Corporation, the worldwide leader in software, services, and solutions that help people and businesses realize their full potential. Microsoft had revenues of US$39.79 billion for the fiscal year ending June, 2005, and employs more than 61,000 people in 102 countries and regions. According to Forbes, Bill Gates is the richest person in the world with a net worth of about US$50 billion, as of March, 2006.

Coreference resolution

Relation extraction

[Bill Gates] co-founder of [Microsoft Corporation]
[Bill Gates] chairman of [Microsoft Corporation]
Contents

- **Part 1**: Overview of text data analysis
- **Part 2**: Preprocessing text data and text representation
- **Part 3**: Text classification, clustering, and topic analysis
- **Part 4**: Word and phrase analysis
- **Part 5**: Information extraction from texts
- **Part 6**: Vietnamese text and language processing
- **Part 7**: Text analysis application topics
Part 6: Vietnamese text and language processing

- Early efforts for Vietnamese NLP
- Vietnamese NLP resources
- Basic Vietnamese NLP tasks
- Vietnamese NLP tools
- The current stage of Vietnamese NLP
- Vietnamese NLP community and events
Early efforts for Vietnamese NLP

Demonstration of VLSP Products

This is the demonstration website of the Vietnamese Language Processing, a major branch of a KC01.01/06-10 national project named Building Basic Resources and Tools for Vietnamese Language and Speech Processing (VLSP). Demonstration list:

- **SP7.2: Vietnamese Machine Readable Dictionary**
  Key people: Vu Xuan Luong, Ho Tu Bao, Nguyen Thi Minh Huyen

- **SP7.3: Vietnamese Treelbank**
  Key people: Nguyen Phuong Thai, Vu Xuan Luong, Nguyen Thi Minh Huyen

- **SP7.4: Vietnamese Bilingual Corpus**
  Key people: Ho Bao Quoc, Cao Hoang Tru

- **SP8.2: Vietnamese Word-Segmentation Program**
  Key people: Nguyen Thi Minh Huyen, Ho Bao Quoc

- **SP8.3: Vietnamese Part-of-Speech Tagger**
  Key people: Phan Xuan Hieu, Le Minh Hoang

- **SP8.4: Vietnamese Chunker**
  Key people: Nguyen Le Minh, Cao Hoang Tru

- **SP8.5: Vietnamese Syntax Parser**
  Key people: Le Thanh Huong, Nguyen Phuong Thai

This website is developed by Nguyen Viet Cuong and Nguyen Le Minh (JAIST). Vietnamese language processing tools and samples of data are provided by the previous groups. The website uses the following open source tools: PHP, MySQL, Smarty, Snoopy, WZToolTip, Mudim, Zapatec.
Vietnamese NLP resources

- Vietnamese machine readable dictionary
  - 35,000 Vietnamese contemporary words with morphological, syntactic, and semantic information
  - Adopt international standard format for computer dictionaries

- Vietnamese treebank (different versions) – like English Penn treebank format
  - 70,000 word-segmented sentences
  - 10,000 part-of-speech tagged sentences
  - 10,000 syntactic trees

- English-Vietnamese bilingual corpus
  - 80,000 sentence pairs in economics-social topics
  - 20,000 sentence pairs in information technology topic

- Vietnamese WordNet

- Vietnamese dependency parsing corpus

- Vietnamese sentiment analysis corpora
Basic Vietnamese NLP tasks

- Sentent segmentation
- Word segmentation
- Part-of-speech tagging
- Shallow parsing (phrase chunking)
- Named entity recognition
- Syntactic parsing
- Dependency parsing
Vietnamese NLP tools

- JVnTextPro: A Java-based Vietnamese Text Processing Tool
  - Sentence segmentation, word segmentation, part-of-speech tagging

- VNU-HUS Vietnamese NLP tool kits
  - [http://mim.hus.vnu.edu.vn/phuonglh/softwares](http://mim.hus.vnu.edu.vn/phuonglh/softwares)
  - vnTokenizer: Vietnamese word segmentation (Java)
  - vnTagger: Vietnamese part-of-speech tagger (Java)
  - Vn.vitk: Vietnamese word segmentation, POS tagger, dependency parser (Java)
  - Ai.vitk.ner: Vietnamese named entity recognition (Scala)

- VLSP tools
  - [https://vlsp.hpda.vn/demo/?page=resources](https://vlsp.hpda.vn/demo/?page=resources)
  - Vietnamese chunker, syntactic parser
Vietnamese NLP tools (2)

- VnCoreNLP (Python)
  - https://github.com/vncorenlp/VnCoreNLP
  - Word segmentation, POS tagging, named entity recognition
  - Dependency parsing

- NLP-progress
  - https://nlpprogress.com/vietnamese/vietnamese.html
  - Updating progress on Vietnamese NLP research and development
The current stage of Vietnamese NLP (incomplete)

- Word segmentation
- Part-of-speech tagging
- Named entity recognition
- Machine translation: English-to-Vietnamese translation
- Dependency parsing
- Opinion mining and sentiment analysis
## Word segmentation – incomplete list

<table>
<thead>
<tr>
<th>Model</th>
<th>F1</th>
<th>Paper</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>VnCoreNLP-RDRsegmenter (2018)</td>
<td>97.90</td>
<td>A Fast and Accurate Vietnamese Word Segmenter</td>
<td>Official</td>
</tr>
<tr>
<td>UETsegmenter (2016)</td>
<td>97.87</td>
<td>A hybrid approach to Vietnamese word segmentation</td>
<td>Official</td>
</tr>
<tr>
<td>JVnSegmenter (2006)</td>
<td>97.06</td>
<td>Vietnamese Word Segmentation with CRFs and SVMs: An Investigation</td>
<td></td>
</tr>
<tr>
<td>DongDu (2012)</td>
<td>96.90</td>
<td>Úng dụng phương pháp Pointwise vào bài toán tách từ cho tiếng Việt</td>
<td></td>
</tr>
</tbody>
</table>
### Part-of-speech tagging – incomplete list

<table>
<thead>
<tr>
<th>Model</th>
<th>Accuracy</th>
<th>Paper</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>VnCoreNLP-VnMarMoT (2017)</td>
<td>95.88</td>
<td>From Word Segmentation to POS Tagging for Vietnamese</td>
<td>Official</td>
</tr>
<tr>
<td>BiLSTM-CRF + CNN-char (2016)</td>
<td>95.40</td>
<td>End-to-end Sequence Labeling via Bidirectional LSTM-CNNs-CRF</td>
<td>Official / Link</td>
</tr>
<tr>
<td>BiLSTM-CRF + LSTM-char (2016)</td>
<td>95.31</td>
<td>Neural Architectures for Named Entity Recognition</td>
<td>Link</td>
</tr>
<tr>
<td>BiLSTM-CRF (2015)</td>
<td>95.06</td>
<td>Bidirectional LSTM-CRF Models for Sequence Tagging</td>
<td>Link</td>
</tr>
<tr>
<td>RDRPOSTagger (2014)</td>
<td>95.11</td>
<td>RDRPOSTagger: A Ripple Down Rules-based Part-Of-Speech Tagger</td>
<td>Official</td>
</tr>
</tbody>
</table>
### Named entity recognition – incomplete list

<table>
<thead>
<tr>
<th>Model</th>
<th>F1</th>
<th>Paper</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>VnCoreNLP (2018)</td>
<td>88.55</td>
<td>VnCoreNLP: A Vietnamese Natural Language Processing Toolkit</td>
<td>Official</td>
</tr>
<tr>
<td>BiLSTM-CRF + CNN-char (2016)</td>
<td>88.28</td>
<td>End-to-end Sequence Labeling via Bidirectional LSTM-CNNs-CRF</td>
<td>Official / Link</td>
</tr>
<tr>
<td>BiLSTM-CRF + LSTM-char (2016)</td>
<td>87.71</td>
<td>Neural Architectures for Named Entity Recognition</td>
<td>Link</td>
</tr>
<tr>
<td>BiLSTM-CRF (2015)</td>
<td>86.48</td>
<td>Bidirectional LSTM-CRF Models for Sequence Tagging</td>
<td>Link</td>
</tr>
</tbody>
</table>
# Machine translation: English-to-Vietnamese – incomplete list

<table>
<thead>
<tr>
<th>Model</th>
<th>BLEU</th>
<th>Paper</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>CVT (2018)</td>
<td>29.6</td>
<td>Semi-Supervised Sequence Modeling with Cross-View Training</td>
<td></td>
</tr>
<tr>
<td>ELMo (2018)</td>
<td>29.3</td>
<td>Deep contextualized word representations</td>
<td></td>
</tr>
<tr>
<td>Transformer (2017)</td>
<td>28.9</td>
<td>Attention is all you need</td>
<td>Link</td>
</tr>
<tr>
<td>Google (2017)</td>
<td>26.1</td>
<td>Neural machine translation (seq2seq) tutorial</td>
<td>Official</td>
</tr>
</tbody>
</table>
# Dependency parsing – incomplete list

<table>
<thead>
<tr>
<th>Model</th>
<th>LAS</th>
<th>UAS</th>
<th>Paper</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Predicted POS</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VnCoreNLP (2018)</td>
<td>70.23</td>
<td>76.93</td>
<td>VnCoreNLP: A Vietnamese Natural Language Processing Toolkit</td>
<td>Official</td>
</tr>
<tr>
<td><strong>Gold POS</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VnCoreNLP (2018)</td>
<td>73.39</td>
<td>79.02</td>
<td>VnCoreNLP: A Vietnamese Natural Language Processing Toolkit</td>
<td>Official</td>
</tr>
<tr>
<td><strong>Gold POS</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BiLSTM graph-based parser</td>
<td>73.17</td>
<td>79.39</td>
<td>Simple and Accurate Dependency Parsing Using Bidirectional LSTM Feature Representations</td>
<td>Official</td>
</tr>
</tbody>
</table>
## Dependency parsing – incomplete list (2)

<table>
<thead>
<tr>
<th>Gold POS</th>
<th>BiLSTM transition-based parser (2016)</th>
<th>72.53</th>
<th>79.33</th>
<th>Simple and Accurate Dependency Parsing Using Bidirectional LSTM Feature Representations</th>
<th>Official</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gold POS</td>
<td>MSTparser (2006)</td>
<td>70.29</td>
<td>76.47</td>
<td>Online large-margin training of dependency parsers</td>
<td></td>
</tr>
<tr>
<td>Gold POS</td>
<td>MaltParser (2007)</td>
<td>69.10</td>
<td>74.91</td>
<td>MaltParser: A language-independent system for datadriven dependency parsing</td>
<td></td>
</tr>
</tbody>
</table>
Vietnamese NLP community and events

- VLSP: Vietnamese Language and Speech Processing
  - Vietnamese language processing
  - Vietnamese speech processing

- VLSP events
  - VLSP-2012 (joint event with IEEE-RIVF-2012); VLSP-2013 (joint event with IEEE-RIVF-2013)
  - VLSP-2015 (joint event with PAKDD-2015); VLSP-2016 (joint event with IEEE RIVF-2016)
  - VLSP-2018 (joint event with CLCLing-2018); VLSP-2019 (will be held with PACLING-2019)

- VLSP shared tasks
  - 2013: word segmentation and POS tagging tasks + shared datasets
  - 2016: named entity recognition and sentiment analysis tasks + shared datasets
  - 2018: named entity recognition and sentiment analysis tasks + shared datasets
Contents

- **Part 1**: Overview of text data analysis
- **Part 2**: Preprocessing text data and text representation
- **Part 3**: Text classification, clustering, and topic analysis
- **Part 4**: Word and phrase analysis
- **Part 5**: Information extraction from texts
- **Part 6**: Vietnamese text and language processing
- **Part 7**: Text analysis application topics
Part 7: Text analysis application topics

- Text information management system
- Opinion mining and sentiment analysis applications
- Text analysis in social media
- Text content matching for contextual advertising
- Text summarization applications
- Text generation applications
- Text understanding and chatbot applications
- Text analysis applications in aviation
“Easy” vs. “difficult” text analysis and NLP applications

Tasks: 
- Classification/retrieval
- Summarization/extraction/topic mining
- Translation/dialogue
- Question answering

Dependency on NLP:

“Easier” and more “workarounds”
Smart text information and management system
Opinion mining and sentiment analysis applications
Text analysis in online social media
ISM – intelligent social media monitoring system
Text content matching for contextual advertising

A solution for "reaching the right person with the right message at the right time"
Can Amazon and Gmail understand us?
Text summarization applications

- Book summarization
- Online news summarization
- Making newsletters
- Email summarization
- Etc.
Text generation applications

- Question-answering
- Virtual assistant/chatbots
- Summarization
- Writing blogs/news articles
Text understanding and chatbot applications
Chatbot platforms

- Bots for Messenger
- Microsoft Bot Framework
- A.L.I.C.E and Pandorabots Platform
- Howdy Botkit
- Rebot.me
AlaaS and NLPaaS

wit.ai
Now part of Facebook

Watson Conversation

Microsoft LUIS

Active.ai

Google Cloud Platform

Google Natural Language API
Text understanding for building virtual assistants & chatbots

Transforming natural language sentences into executable forms

**Input:** *turn it off please*
- intent-name = turn-off
- domain = device-control
- object = it
- context = {current-object = “television”}

**Input:** *send Viber message to Nga saying we have party tonight*
- intent-name = send-message
- domain = message
- contact = Nga
- means = Viber
- message-content = “we have party tonight”
# Simple intent identification in spoken texts

<table>
<thead>
<tr>
<th>Vietnamese spoken language command</th>
<th>Intent (a::f)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ngã tự tây sơn chưa bóc ở đâu  (where is tay son chưa bóc intersection)</td>
<td>map::locate</td>
</tr>
<tr>
<td>tìm đường tự hà đông đến 88 lange hà  (find direction from ha dong to 88 lang ha)</td>
<td>map::find-direction</td>
</tr>
<tr>
<td>gọi số 0903206714 (call number 0903206714)</td>
<td>phone::call</td>
</tr>
<tr>
<td>đặt lịch họp với ibm 9 giờ 15 sáng thứ tư tuần sau (arrange a meeting with ibm at quarter pass 9 next wednesday morning)</td>
<td>calendar::set</td>
</tr>
<tr>
<td>vào trang dẫn trí chăm com chăm vn (open page dan tri dot com dot vn)</td>
<td>browser::open</td>
</tr>
<tr>
<td>mò bài ha trang (play song ha trang)</td>
<td>music::open</td>
</tr>
<tr>
<td>thời tiết vùng tàu ngà kia (the weather in vung tau the day after tomorrow)</td>
<td>weather::query</td>
</tr>
<tr>
<td>hôm nay ngày bao nhiêu âm lịch (what lunar date is today)</td>
<td>calendar::query</td>
</tr>
<tr>
<td>gửi số điện thoại của yên cho đường vu (send phone number of yen to duong vu)</td>
<td>contact::share</td>
</tr>
<tr>
<td>mở twitter (open twitter)</td>
<td>other-app::open</td>
</tr>
</tbody>
</table>
Simple intent’s parameters identification

[ngã tư sỏ] ở đâu (where is [so intersection])
dánh thức lúc [7 giờ kém 15 phút sáng] (wake up at [quarter to 7 o’clock am])
gọi số [0903206714] (call number [0903206714])
vào trang [đàn trí chạm com chắm vn] (open page [dan tri dot com dot vn])
dặt lịch họp với ibm lúc [9 giờ 15 sáng thứ tư tuần sau]
(arrange a meeting with ibm at [quarter pass 9 wednesday morning next week])
tìm đường từ [ga hà nội] đến [88 lang hà]
(find direction from [hanoi station] to [88 lang ha])
thời tiết [vùng tàu] [ngày kia]
(weather in [vung tau] [the day after tomorrow])
âm lịch [hôm nay] ngày bao nhiêu (what lunar date is [today])
gửi email cho [anh nam] (send email to [anh nam])
mở [skype] (open [skype])
gửi số điện thoại của [yến] cho [duong vu]
(send the phone number of [yen] to [duong vu])
Building assistants/chatbots: major issues

- Virtual assistants vs. chatbots
- Shallow or deep intent understanding?
- Intent’s parameter/argument identification (complex NER)
- Speech/dialogue act identification
- Context understanding
- Conversation/dialogue management
- Learnability
- Natural language generation
Text analysis applications in aviation

- Question answering and chatbots
- Passenger’s email filtering and routing
  - Classifying clients/passengers’ email into different business sections
- Passenger’s feedback sentiment analysis
  - Quantify passengers’ comments
  - Improving quality of services
Question answering and chatbots for aviation/airline

- Question answering
  - FAQs (e.g., answering aviation/airline information)

- Utility chatbots
  - Search and check flights
  - Search and find tickets
  - Online checkin
  - Etc.
Passenger’s feedback sentiment analysis

- **Aspects:**
  - Inflight-service
  - Attendant
  - Food-drink
  - Ground-service
  - Luggage
  - Aircraft-condition
  - Entertainment

- **Type of sentences:**
  - Suggestion (sugg)
  - Complaint (comp)
  - Recognition (recg)
  - Other (othe)
Lecture summary

- **Part 1**: Overview of text data analysis
- **Part 2**: Preprocessing text data and text representation
- **Part 3**: Text classification, clustering, and topic analysis
- **Part 4**: Word and phrase analysis
- **Part 5**: Information extraction from texts
- **Part 6**: Vietnamese text and language processing
- **Part 7**: Text analysis application topics