Text Analytics

MODULE 3: GET THE TEXT DATA READY FOR ANALYSIS

Dr. Fan Zhenzhen
Institute of Systems Science
National University of Singapore
Email: zhenzhen@nus.edu.sg

© 2015-2021 NUS. The contents contained in this document may not be reproduced in any form or by any means, without the written permission of ISS, NUS, other than for the purpose for which it has been supplied.

© 2015-2021 National University of Singapore. All Rights Reserved

Objectives

At the end of this module, you can:

- Define the metadata and corpus to be imported into TA repository
- Identify preprocessing steps of text data typically required for TA tasks
- Develop term-document frequency matrix to enable lookup of text and documents within the corpus

© 2015-2021 National University of Singapore. All Rights Reserved

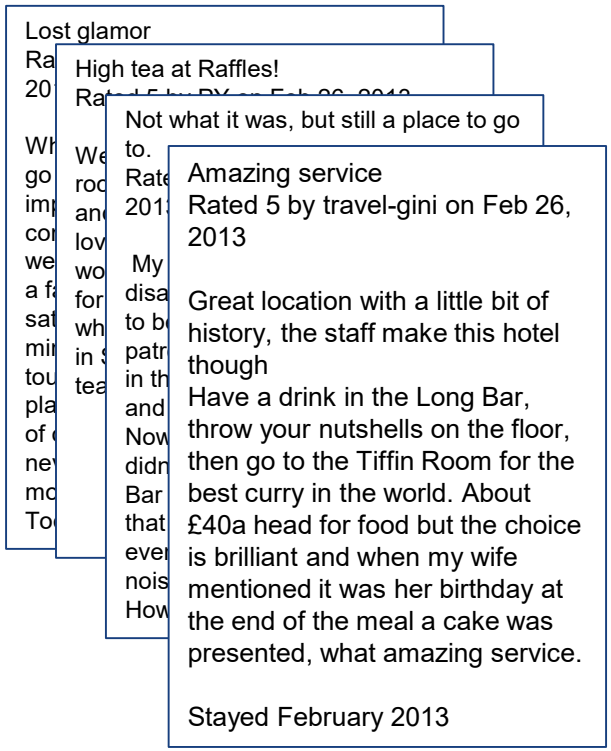
Outline

- Overview of task
- Text preprocessing
- From text to word – tokenization, stemming, stopword removal
- Convert text to term-document frequency matrix (indexing)

© 2015-2021 National University of Singapore. All Rights Reserved

The whole task here is…

Documents

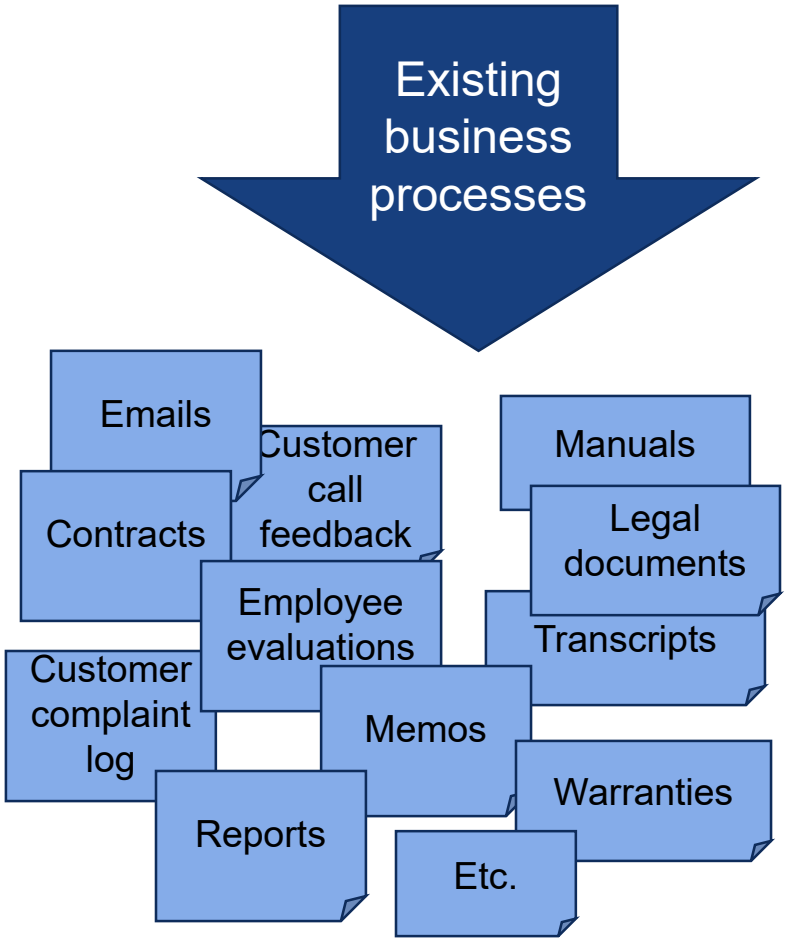


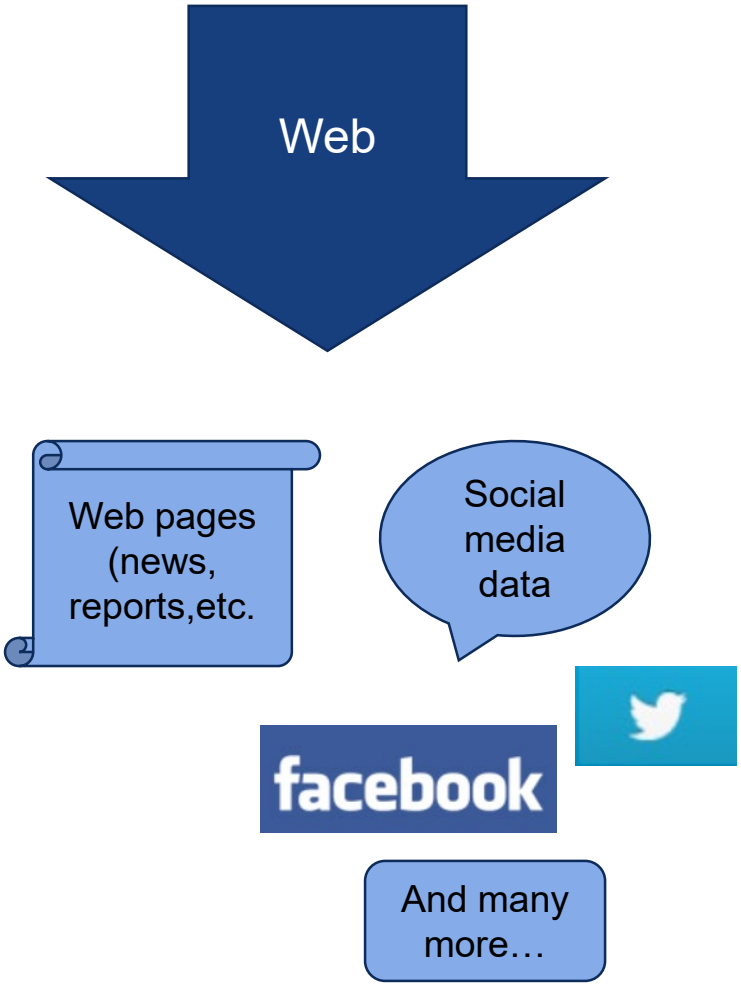
Term Document Matrix

	amazing	service	lost	glamour	disappoint	brilliant	super	expensive	noisy	…
Doc1	1	1	0	0	0	1	0	0	0	
Doc2	0	0	1	1	1	0	0	1	0	
Doc3	0	0	0	1	0	0	1	0	0	
Doc4	0	0	0	0	2	0	0	1	1	
…										

© 2015-2021 National University of Singapore. All Rights Reserved

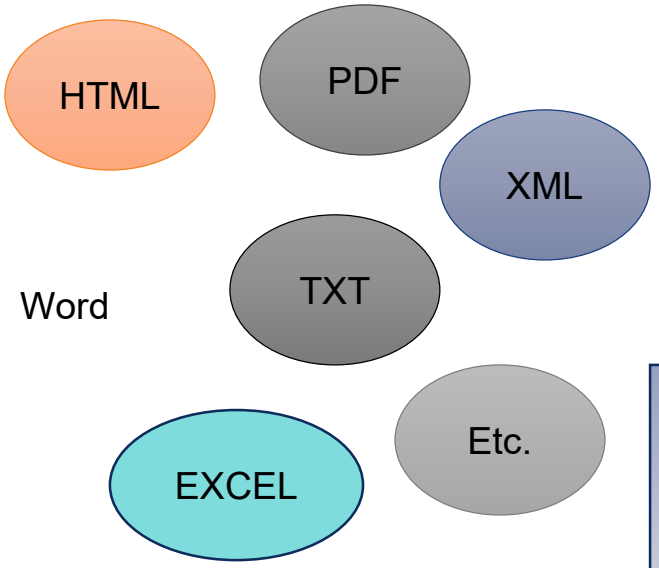
Sources of Text Data

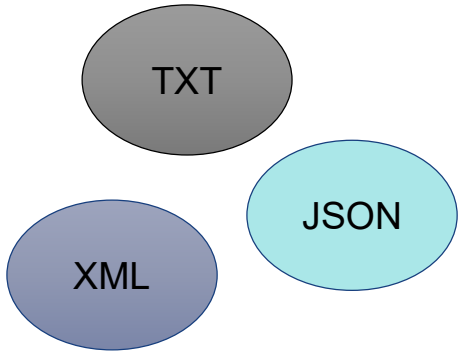




© 2015-2021 National University of Singapore. All Rights Reserved

File Preprocessing





—Delete formatting tags
—Remove special characters
—Detect and label different zones
—Determine sentence/paragraph boundaries

Most TA tools provide functionality of importing text from some common formats.

© 2015-2021 National University of Singapore. All Rights Reserved

In a pure text file…

Amazing service

Rated 5 by travel-gini on Feb 26, 2013

Great location with a little bit of history, the staff make this hotel though
Have a drink in the Long Bar, throw your nutshells on the floor, then go to the Tiffin Room for the best curry in the world. About £40a head for food but the choice is brilliant and when my wife mentioned it was her birthday at the end of the meal a cake was presented, what amazing service.

Stayed February 2013

…

© 2015-2021 National University of Singapore. All Rights Reserved

XML as Standard Exchange Format

- Popular in industry and text-processing community
- With XML, we can insert tags onto a text to identify its parts.
  - Eg. <*DOC*>, <*SUBJECT*>, <*TOPIC*>, <*TEXT*>, etc.
  - Such tags are very useful as they allow selection/extraction of the parts to generate features for subsequent mining.
- Many word processors allow documents to be saved as XML format

© 2015-2021 National University of Singapore. All Rights Reserved

What would an XML doc look like?

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<REVIEWS>
    <REVIEW>
        <TITLE>Amazing service</TITLE>
        <RATING>5</RATING>
        <DATE>26/02/2013</DATE>
        <BY>travel-gini</BY>
        <CONTENT>Great location with a little bit of history, the staff make this hotel though
    Have a drink in the Long Bar, throw your nutshells on the floor, then go to the Tiffin Room for the best
    curry in the world. About £40a head for food but the choice is brilliant and when my wife mentioned it
    was her birthday at the end of the meal a cake was presented, what amazing service. Stayed
    February 2013 </CONTENT>
    </REVIEW>
    …
</REVIEWS>
```


© 2015-2021 National University of Singapore. All Rights Reserved

Or as JSON format

{

"title": "Amazing service",

"rating": 5,

"date": "26/02/2013",

"by": "travel-gini",

"content": "Great location with a little bit of history, the staff make this hotel though Have a drink in the Long Bar, throw your nutshells on the floor, then go to the Tiffin Room for the best curry in the world. About £40a head for food but the choice is brilliant and when my wife mentioned it was her birthday at the end of the meal a cake was presented, what amazing service. Stayed February 2013"

}

© 2015-2021 National University of Singapore. All Rights Reserved

From Text to Words

© 2015-2021 National University of Singapore. All Rights Reserved

Tokenization

- To break a stream of characters into tokens

Great location with a little bit of history.

Great | location | with | a | little | bit | of | history | .

- This is known as unigram model – every token is a single word.
- There are also bigram, trigram models, where each token is composed of two/three words.

Great location | location with | with a | a little | little bit

bit of | of history | history .

© 2015-2021 National University of Singapore. All Rights Reserved

Tokenization Challenges

- Tokenization is done by identifying token delimiters
  - Whitespace characters such as ***space, tab, newline***
  - Punctuation characters like ***( ) < > ! ? “ ”***
  - Other characters ***. , : - ‘ ’*** etc.
- It seems simple, but…
  - ***. , :*** between numbers are part of the number

    12.34 12,345 12:34
  - **.** can be part of an abbreviation or end of a sentence

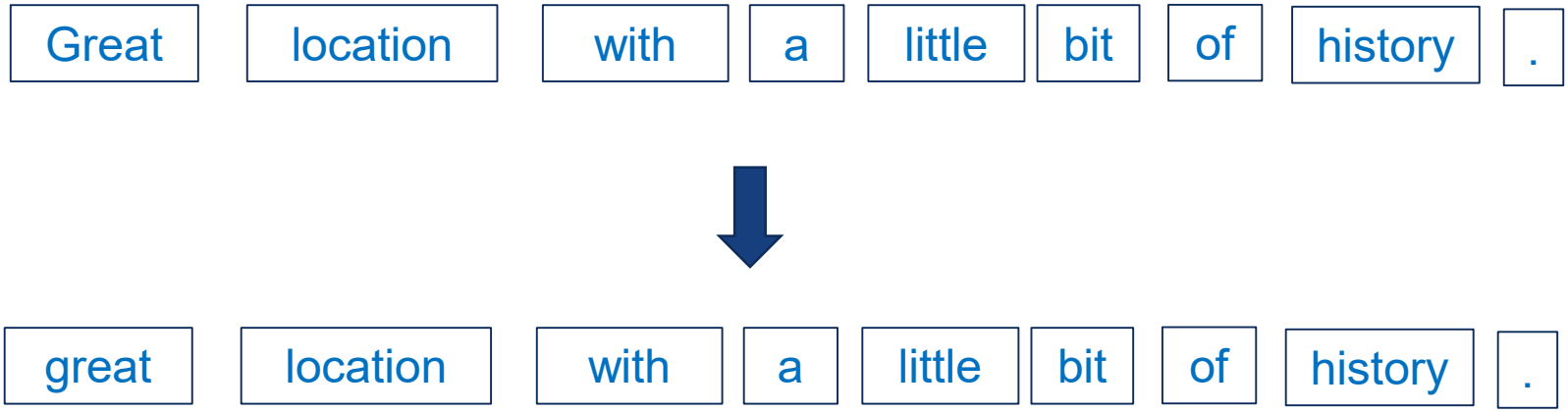
    U.S.A. Dr.
  - **’** can be a closing internal quote, indicate a possessive, or be part of another token

    My friend’s isn’t

© 2015-2021 National University of Singapore. All Rights Reserved

Case Lowering

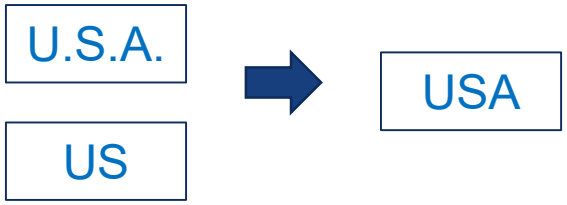
- Also known as **case normalization**, to convert all tokens to lower case to remove the variation of words due to case differences.



© 2015-2021 National University of Singapore. All Rights Reserved

Stemming/Lemmatization

- A word may come in varied forms and therefore need to be converted into a standard form
  - **Inflectional** stemming (no change of POS) -- **Lemmatization**
  - **Derivational** Stemming (with change of POS)
  - Other normalisation (including case normalisation)



- Stemming can reduce the number of distinct features in a text corpus and increase the frequency of occurrence of some individual features.

© 2015-2021 National University of Singapore. All Rights Reserved

How much stemming should be done?

- An inflectional stemmer needs to be partly rule-based and partly dictionary-based.
- Derivational stemming is more aggressive and therefore can reduce the number of features in a corpus drastically. However meaning might be lost in the stemming process.

Too aggressive stemming can result in loss of meaning and non-legitimate words without the support of a dictionary.

```
[[6]]
battery life portability accessories style

[[7]]
   ability  store    music     ability  create playlists

[[8]]
portability capacity sound quality durability
```

```
[[6]]
batteri life portabl accessori style

[[7]]
   abil  store    music     abil  creat playlist

[[8]]
portabl capac sound qualiti durabl
```


© 2015-2021 National University of Singapore. All Rights Reserved

Stemmers

- Some well-known stemming algorithms for English
  - Lovins Stemmer by Julie Beth Lovins, 1968
    - single pass, longest-match
    - removing the longest suffix, ensuring the remaining stem is at least 3 characters long
    - reforming the stem through recoding transformations
  - Porter Stemmer by Martin Porter, 1980
    - Widely used, with implementations in various languages available online (C, java, Perl, python, C#, VB, Javascript, Tcl, Ruby, etc.)
  - Snowball by Porter, a framework for writing Stemming algorithms

© 2015-2021 National University of Singapore. All Rights Reserved

Stopword Removal

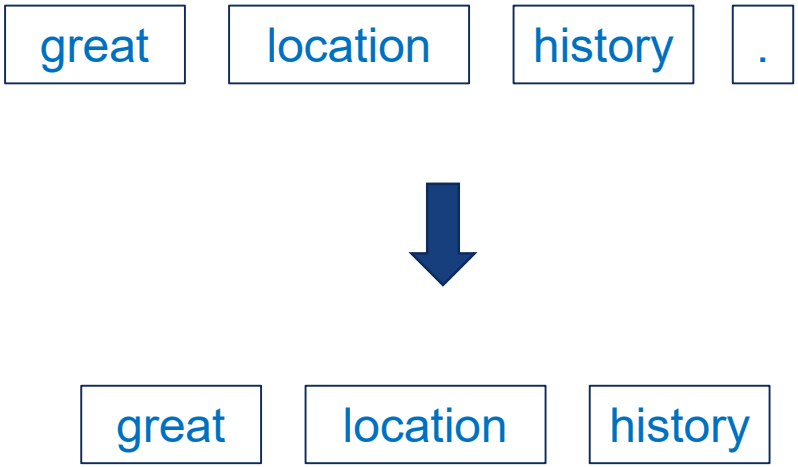
- Some words are extremely common. They appear in almost all documents and carry little meaning. They are of limited use in text analytics applications.
  - Functional words (conjunctions, prepositions, determiners, or pronouns) like *the, of, to, and, it*, etc.
  - A stopword list can be constructed to exclude them from analysis.
  - **Depending on the domain**, other words may need to be included in the stopword list.

great	location	with	a	little	bit	of	history	.

© 2015-2021 National University of Singapore. All Rights Reserved

Further clean-up

- Punctuation removal
- Number removal, etc.



© 2015-2021 National University of Singapore. All Rights Reserved

Indexing

© 2015-2021 National University of Singapore. All Rights Reserved

Indexing

- Many text mining applications are based on vector representation of documents (*term-document matrix* or *document-term matrix*) using "bag-of-words" approach

$$\begin{array}{c|cccc} & T_1 & T_2 & \cdots & T_t \\ D_1 & w_{11} & w_{21} & \cdots & w_{t1} \\ D_2 & w_{12} & w_{22} & \cdots & w_{t2} \\ \vdots & \vdots & \vdots & & \vdots \\ \vdots & \vdots & \vdots & & \vdots \\ D_n & w_{1n} & w_{2n} & \cdots & w_{tn} \end{array}$$

T: term
D: document
w: **weight** of the term

- Usually only content words (adjectives, adverbs, nouns, and verbs) are used as vector features.

© 2015-2021 National University of Singapore. All Rights Reserved

Term Weighting

- Binary
  - 0 or 1, simply indicating whether a word has occurred in the document (but that's not very helpful).
- Frequency-based
  - *term frequency*, the frequency of words in the document, which provides additional information that can be used to contrast with other documents.

	amazing	service	lost	glamour	disappoint	brilliant	super	expensive	noisy	…
Doc1	1	1	0	0	0	1	0	0	0	
Doc2	0	0	1	1	1	0	0	1	0	
Doc3	0	0	0	1	0	0	1	0	0	
Doc4	0	0	0	0	2	0	0	1	1	
…										

© 2015-2021 National University of Singapore. All Rights Reserved

Frequent Word List

- With frequency-based TDM, a list of words and their frequencies in the corpus can be generated
  - *Global frequency* – how many times a word appears in the corpus
  - *Document frequency* – how many unique documents contain the word

- This list, sorted by frequency, can give us a rough idea of what the corpus is about.

- Word Cloud is a nice visualization of such information.

© 2015-2021 National University of Singapore. All Rights Reserved

Word Cloud: another example

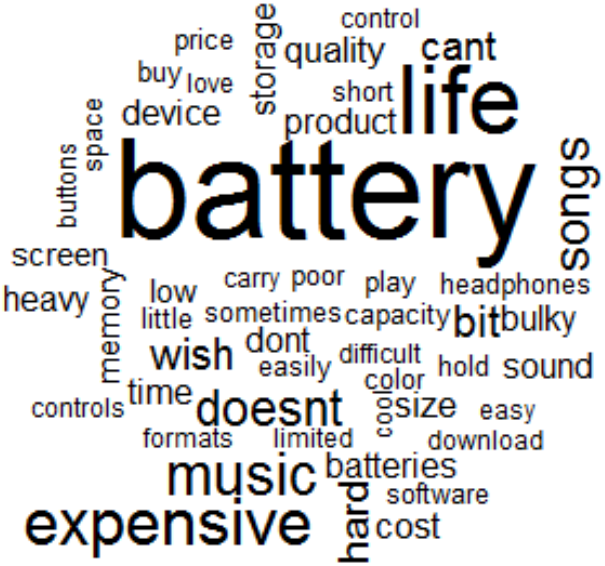
- Generated from http://worditout.com/word-cloud/make-a-new-one

© 2015-2021 National University of Singapore. All Rights Reserved

Contrasting Two Groups

- "What do you like most…"

- "What do you like least…"



© 2015-2021 National University of Singapore. All Rights Reserved

Other Weighting Methods

- Normalized frequency
  - To deal with varied document length, since a long document definitely has more occurrences of terms than a short document

$$normalized\_frequency = \frac{frequency\ of\ a\ term\ in\ a\ document}{total\ number\ of\ terms\ in\ the\ document}$$

- ***tf-idf***
  - To modify the frequency of a word in a document by the perceived importance of the word(the *inverse document frequency)*, widely used in information retrieval
    - When a word appears in many documents, it's considered unimportant.
    - When the word is relatively unique and appears in few documents, it's important.

© 2015-2021 National University of Singapore. All Rights Reserved

tf-idf Indexing

- *tf-idf* weighting :

$$tf\text{-}idf_{t,d} = tf_{t,d} * idf_t$$

  - $tf_{t,d}$: *term frequency* – number of occurrences of term t in document d
  - idf_t : *inverted document frequency* of term t

$$idf_t = \log \frac{N}{df_t}$$

N : the total number of documents in the corpus

df_t : the document frequency of term t, i.e., the number of documents that contain the term.

© 2015-2021 National University of Singapore. All Rights Reserved

tf-idf Indexing – An Example

Note that in this example, stopwords and very common words are not removed, and terms are not reduced to root terms.

TERM VECTOR MODEL BASED ON $w_i = tf_i * IDF_i$

Query, Q: "gold silver truck"
D_1: "Shipment of gold damaged in a fire"
D_2: "Delivery of silver arrived in a silver truck"
D_3: "Shipment of gold arrived in a truck"
$D = 3$; $IDF = \log(D/df_i)$

		Counts, tf_i						Weights, $w_i = tf_i * IDF_i$			
Terms	Q	D_1	D_2	D_3	df_i	D/df_i	IDF_i	Q	D_1	D_2	D_3
a	0	1	1	1	3	3/3 = 1	0	0	0	0	0
arrived	0	0	1	1	2	3/2 = 1.5	0.1761	0	0	0.1761	0.1761
damaged	0	1	0	0	1	3/1 = 3	0.4771	0	0.4771	0	0
delivery	0	0	1	0	1	3/1 = 3	0.4771	0	0	0.4771	0
fire	0	1	0	0	1	3/1 = 3	0.4771	0	0.4771	0	0
gold	1	1	0	1	2	3/2 = 1.5	0.1761	0.1761	0.1761	0	0.1761
in	0	1	1	1	3	3/3 = 1	0	0	0	0	0
of	0	1	1	1	3	3/3 = 1	0	0	0	0	0
silver	1	0	2	0	1	3/1 = 3	0.4771	0.4771	0	0.9542	0
shipment	0	1	0	1	2	3/2 = 1.5	0.1761	0	0.1761	0	0.1761
truck	1	0	1	1	2	3/2 = 1.5	0.1761	0.1761	0	0.1761	0.1761

http://www.miislita.com/term-vector/term-vector-3.html

© 2015-2021 National University of Singapore. All Rights Reserved

Alternative Representation of TDM

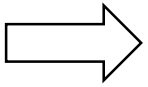
- The resulting term document matrix is expected to have most of the values to be zero, since typically a document will only contain a small subset of the vocabulary in a corpus

```
<<DocumentTermMatrix (documents: 1000, terms: 17887)>>
Non-/sparse entries: 92858/17794142
Sparsity           : 99%
Maximal term length: 56
Weighting          : term frequency (tf)
```

- It saves memory to store the matrix as a set of sparse vectors, where a row is represented by a list of pairs, (ColumnNumber, Value)

Matrix

0	5	2	0
4	0	0	0
3	1	0	6

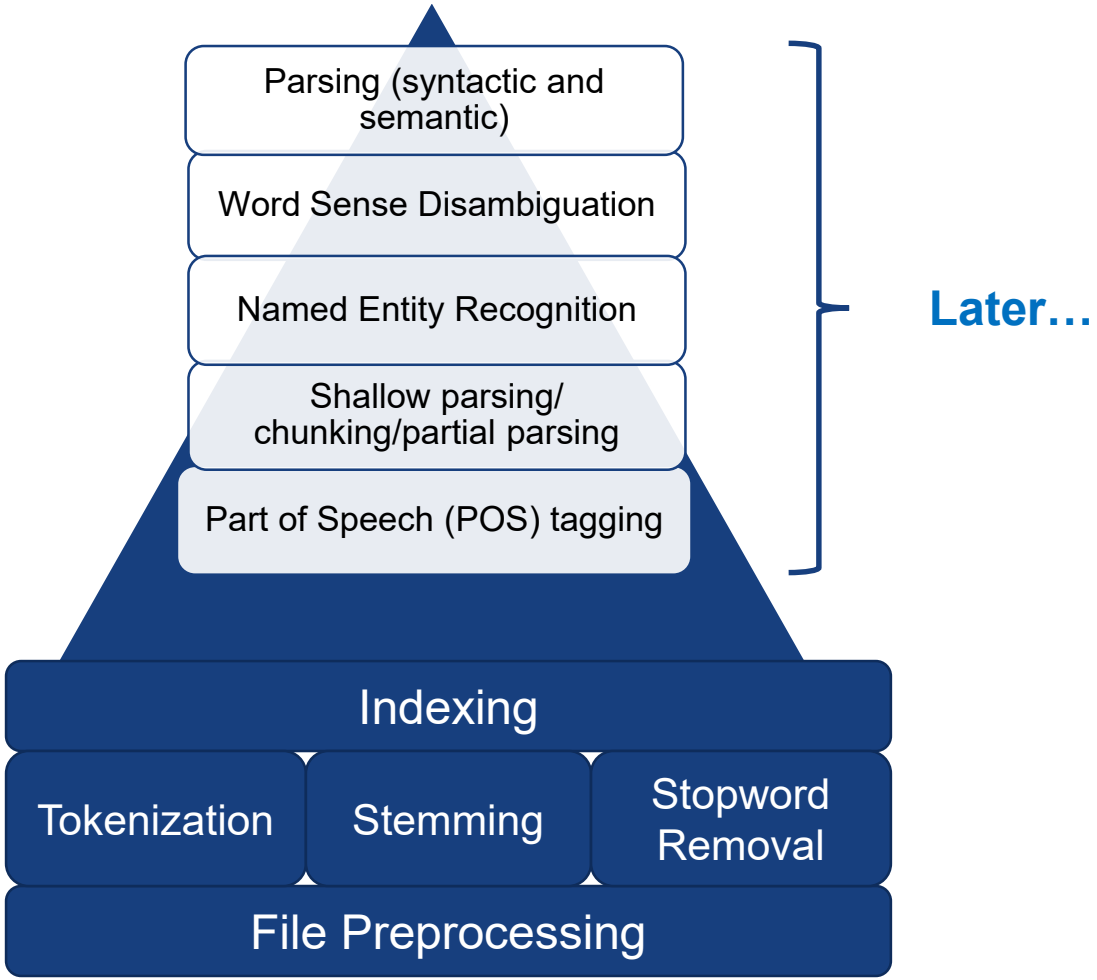


Sparse Vectors

(2, 5) (3, 2)
(1, 4)
(1, 3) (2, 1) (4, 6)

© 2015-2021 National University of Singapore. All Rights Reserved

Summary



© 2015-2021 National University of Singapore. All Rights Reserved

Reference & Resources

- Jurafsky, Dan. *Speech & language processing*. Pearson Education India, 2000. (continuously updated)
- Weiss, Indurkhya, & Zhang. Chapter 2 “From Textual Information to Numerical Vectors”, *Fundamentals of Predictive Text Mining*, Springer, 2010.
- List of online word cloud generators
  - http://www.techlearning.com/default.aspx?tabid=67&entryid=364

© 2015-2021 National University of Singapore. All Rights Reserved