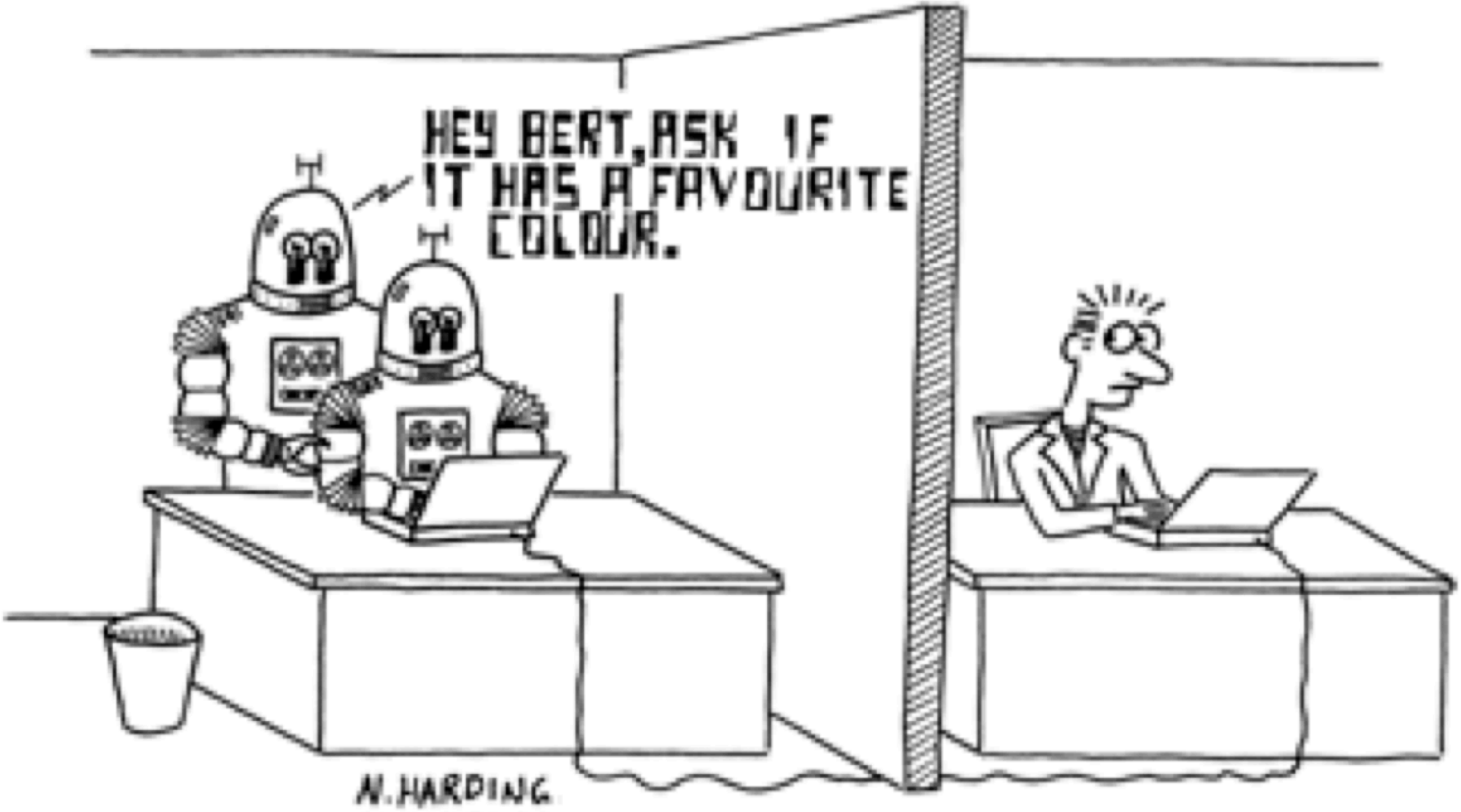
Technology Fundamentals for Analytics Lab

Jason Kuruzovich

Agenda

Introduction to Text Analysis


HEY BERT, ASK IF IT HAS A FAVOURITE COLOUR.
N. HARDING

The Turing test is a test of a machine's ability to exhibit intelligent behavior equivalent to, or indistinguishable from, that of a human.

A computer just passed the Turing Test in landmark trial

A 140

By Terrence McCoy June 9 Follow @terrence_mc

Can machines think?

In 1950, famed London scientist Alan Turing, considered one of the fathers of artificial intelligence, published a paper that put forth that very question. But as quickly he asked the question, he called it "absurd." The idea of thinking was too difficult to define. Instead, he devised a separate way to quantify mechanical "thinking."

Alan Turing from archive of papers relating to the development of computing at the National Physical Laboratory between the late 1940s and the early 1970s. (Science Museum, London/SSPL)

"I shall replace the question by another, which is closely related to it and is expressed in relatively unambiguous words," he wrote in the study that some say represented the "beginning" of artificial intelligence. "The new form of the problem can be described in terms of a game which we call the 'imitation game.'"

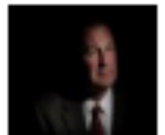
Most Read National

1 The giant stone circles in the Middle East no one can explain

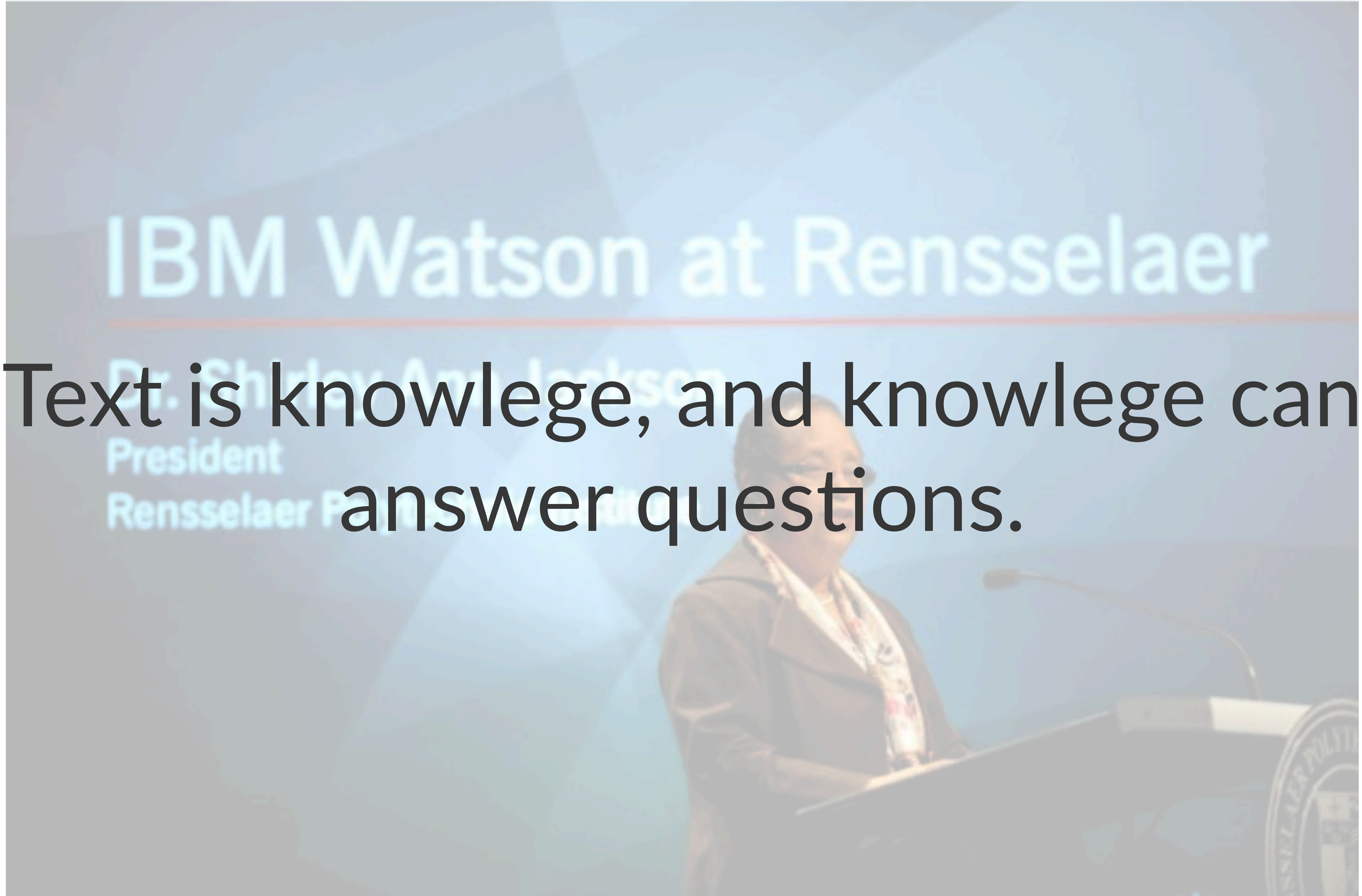
2 Terminally ill Brittany Maynard takes her own life

3 D.C. attorney found bound, gagged and strangled to death in Dominican Republic

4 A father's scars: For Va.'s Creigh Deeds, tragedy brings unending questions




IBM Watson at Rensselaer
Dr. Shirley Ann Jackson
President
Rensselaer Polytechnic Institute

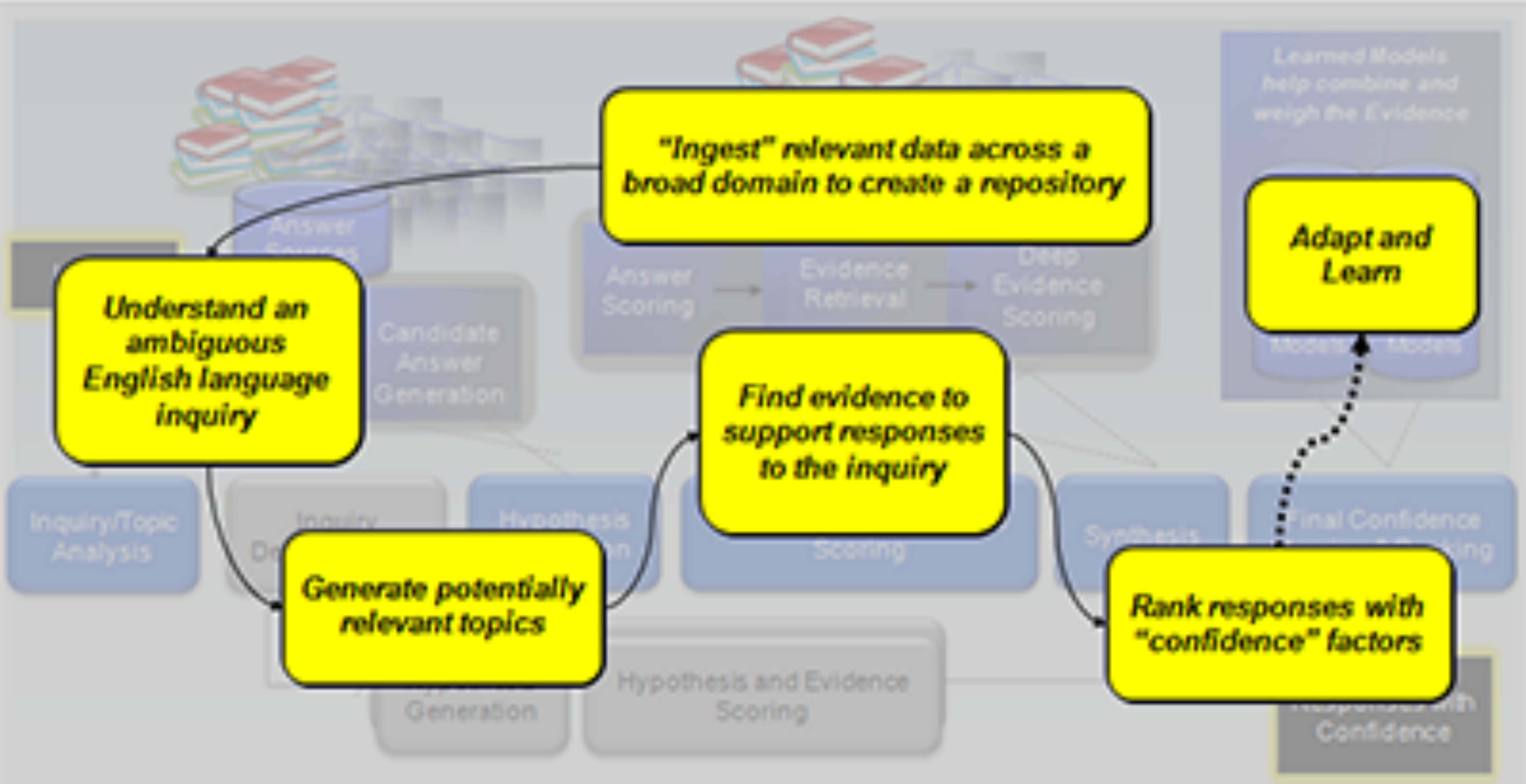


Text is knowlege, and knowlege can answer questions.

What is Watson?

The Science Behind an Answer

Imagine if you could read, understand, and recall all relevant data…



Big Data Problem: Could Fake Reviews Kill Amazon?

http://www.datasciencecentral.com/profiles/blogs/could-fake-reviews-kill-amazon

Cross Industry Standard Process for Data Mining (CRISP-DM; Shearer, 2000),

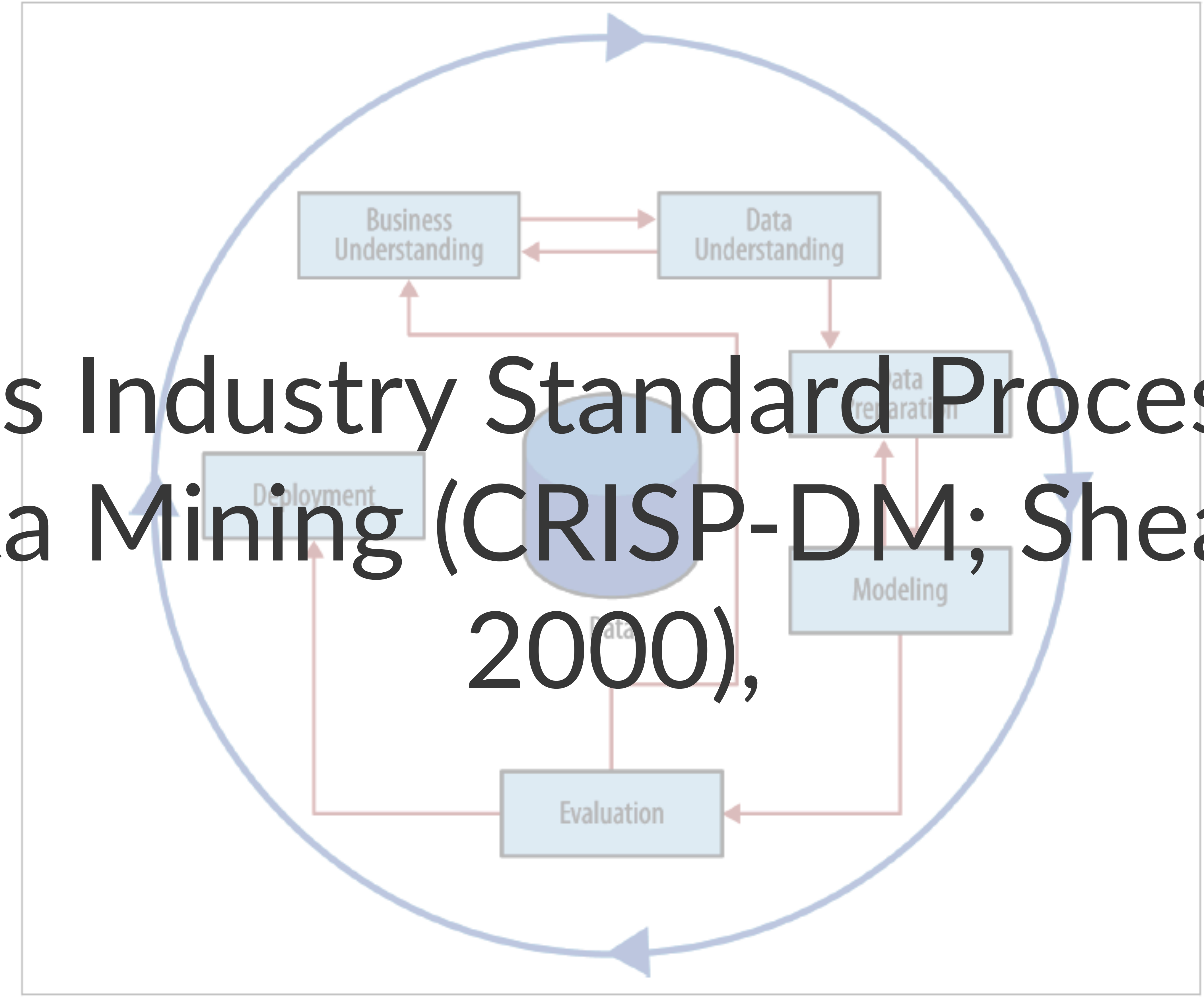


Figure 2-2. The CRISP data mining process.

Stages of Model Development

Pay attention we will use this as a framework

1. Data understanding
2. Data preparation
3. Modeling
4. Evaluation
5. Deployment (DDD)
5. Business Understanding

1. Data Understanding

-A text mining analyst typically starts with a set of highly heterogeneous input texts.
-How is the text formatted?
-What types of text data is present?
-What should be removed because it is irrelevant?
-What types of analyses are likely to be useful?

2. Data preparation

- Stopword removal
- Stemming procedures
- Lemmatisation
- (misc) Remove whitespace, Remove punctuation
- Creation of Corpus
- Create Term Document Matrix
- Create N-gram features

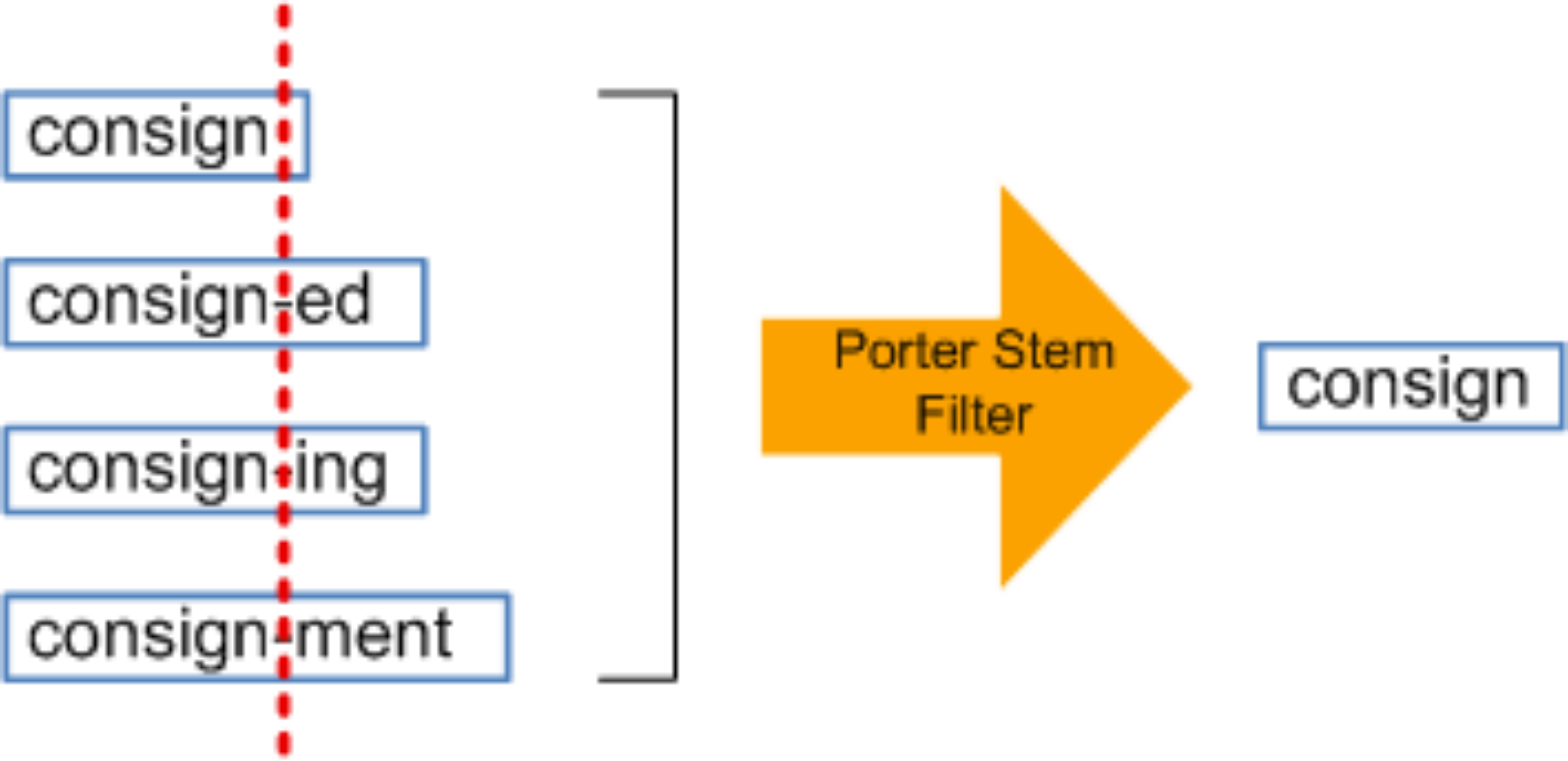
Stop Words

In computing, stop words are words which are filtered out before or after processing of natural language data (text).
There is not one definite list of stop words which all tools use and such a filter is not always used. Some tools specifically avoid removing them to support phrase search.

["a","able","about","above","abst","accordance","according","accordingly","across","act","actually",
"added","adj","adopted","affected","affecting","affects","after","afterwards","again","against","ah","all",
"almost","alone","along","already","also","although","always","am","among","amongst","an","and","announce",
"another","any","anybody","anyhow","anymore","anyone","anything","anyway","anyways","anywhere",
"apparently","approximately","are","aren","arent","arise","around","as","aside","ask","asking","at","auth",
"available","away","awfully","b","back","be","became","because","become","becomes","becoming","been",
"before","beforehand","begin","beginning","beginnings","begins","behind","being","believe","below","beside",
"besides","between","beyond","biol","both","brief","briefly","but","by","c","ca","came","can","cannot","can't",
"cause","causes","certain","certainly","co","com","come","comes","contain","containing","contains","could",
"couldnt","d","date","did","didn't","different","do","does","doesn't","doing","done","don't","down","downwards",
"due","during","e","each","ed","edu","effect","eg","eight","eighty","either","else","elsewhere","end","ending",
"enough","especially","et","et-
al","etc","even","ever","every","everybody","everyone","everything","everywhere","ex","except","f",
"far","few","ff","fifth","first","five","fix","followed","following","follows","for","former","formerly","forth","found",
"four","from","further","furthermore","g","gave","get","gets","getting","give","given","gives","giving","go","goes",
"gone","got","gotten","h","had","happens","hardly","has","hasn't","have","haven't","having","he","hed","hence",
"her","here","hereafter","hereby","herein","heres","hereupon","hers","herself","hes","hi","hid","him","himself","his
", "hither","home","how","howbeit","however","hundred","i","id","ie","if","i'll","im","immediate","immediately",
"importance","important","in","inc","indeed","index","information","instead","into","invention","inward","is","isn'
t", "it","itd","it'll","its","itself","i've","j","just","k","keep","
keeps","kept","keys","kg","km","know","known","knows","l","largely","last","lately","later","latter","latterly",

Stemming

Stemming is the term used in linguistic morphology and information retrieval to describe the process for reducing inflected (or sometimes derived) words to their word stem, base or root form—generally a written word form.


consign
consign-ed
consign-ing
consign-ment
Porter Stem Filter
consign

Lemmatisation (or lemmatization)

Lemmatisation in linguistics is the process of grouping together the different inflected forms of a word so they can be analysed as a single item.

For example, in English, run, runs, ran, and running all correspond to the lemma run.

Text Corpus

In linguistics, a corpus (plural corpora) or text corpus is a large and structured set of texts (nowadays usually electronically stored and processed). They are used to do statistical analysis and hypothesis testing, checking occurrences or validating linguistic rules within a specific language territory.

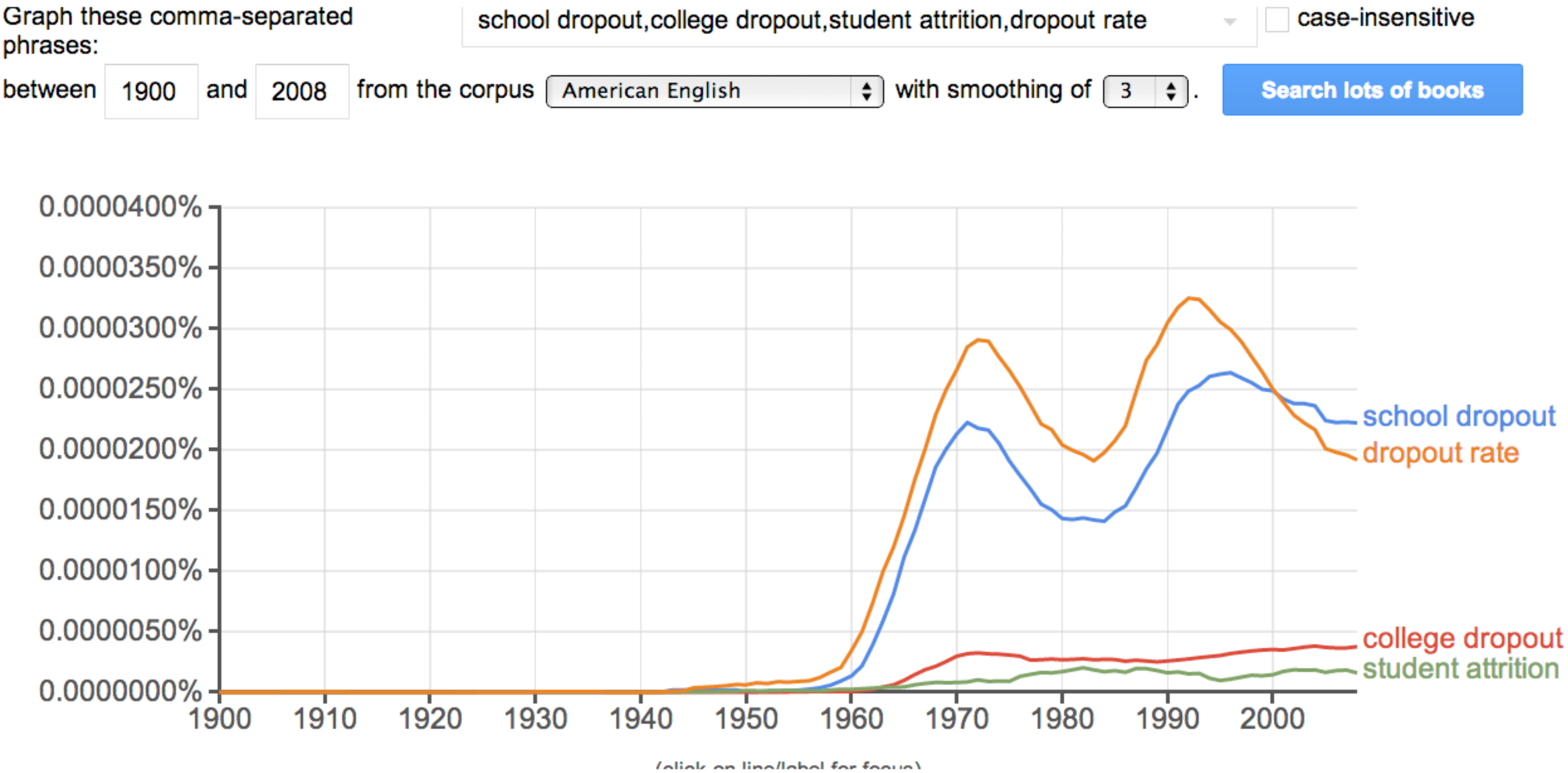
A document-term matrix or term-document matrix is a mathematical matrix that describes the frequency of terms that occur in a collection of documents. In a document-term matrix, rows correspond to documents in the collection and columns correspond to terms

D1 = "I like databases"
D2 = "I hate databases",
then the document-term matrix would be:

	I	like	hate	databases
D1	1	1	0	1
D2	1	0	1	1

N-gram

In the fields of computational linguistics and probability, an n-gram is a contiguous sequence of n items from a given sequence of text or speech. The items can be phonemes, syllables, letters, words or base pairs according to the application. The n-grams typically are collected from a text or speech corpus.


Graph these comma-separated phrases:
school dropout,college dropout,student attrition,dropout rate
case-insensitive
between 1900 and 2008 from the corpus American English with smoothing of 3 .
Search lots of books
0.0000400%
0.0000350%
0.0000300%
0.0000250%
0.0000200%
0.0000150%
0.0000100%
0.0000050%
0.0000000%
1900
1910
1920
1930
1940
1950
1960
1970
1980
1990
2000
school dropout
dropout rate
college dropout
student attrition

Google N-grams

The following is an example of the 4-gram data in this corpus:

serve as the incoming 92
serve as the incubator 99
serve as the independent 794
serve as the index 223
Google gives n-gram

tf–idf

Short for term frequency–inverse document frequency, is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus.

It is often used as a weighting factor in information retrieval and text mining. The tf-idf value increases proportionally to the number of times a word appears in the document, but is offset by the frequency of the word in the corpus, which helps to control for the fact that some words are generally more common than others.

$$\mathrm{tfidf}(t, d, D) = \mathrm{tf}(t, d) \times \mathrm{idf}(t, D)$$

tf = frequency of count

$$\mathrm{idf}(t, D) = \log \frac{N}{|\{d \in D : t \in d\}|}$$

Example of tf–idf [edit]

Suppose we have term frequency tables for a collection consisting of only two documents, as listed on the right, then calculation of tf–idf for the term "this" in document 1 is performed as follows.

Document 1

Term	Term Count
this	1
is	1
a	2
sample	1

Document 2

Term	Term Count
this	1
is	1
another	2
example	3

Tf, in its basic form, is just the frequency that we look up in appropriate table. In this case, it's one.

Idf is a bit more involved:

$$\mathrm{idf}(\mathsf{this}, D) = \log \frac{N}{|\{d \in D : t \in d\}|}$$

The numerator of the fraction is the number of documents, which is two. The number of documents in which "this" appears is also two, giving

$$\mathrm{idf}(\mathsf{this}, D) = \log \frac{2}{2} = 0$$

So tf–idf is zero for this term, and with the basic definition this is true of any term that occurs in all documents.

A slightly more interesting example arises from the word "example", which occurs three times but in only one document. For this document, tf–idf of "example" is:

$$\mathrm{tf}(\mathsf{example}, d_2) = 3$$

$$\mathrm{idf}(\mathsf{example}, D) = \log \frac{2}{1} \approx 0.3010$$

$$\mathrm{tfidf}(\mathsf{example}, d_2) = \mathrm{tf}(\mathsf{example}, d_2) \times \mathrm{idf}(\mathsf{example}, D) = 3 \times 0.3010 \approx 0.9030$$

(using the base 10 logarithm).

Sample Text Corpus

Text annotations can enable reasoning engines

http://www.nactem.ac.uk/resources.php

Text Processing with R

Data Preparation

OLD SCHOOL

With gsub and regular expressions

Processing Text with gsub

```
# Remove specific reference
tweets.text <- gsub("rt", "", tweets.text)

# Replace @UserName
tweets.text <- gsub("@\\w+", "", tweets.text)

# Remove punctuation
tweets.text <- gsub("[[:punct:]]", "", tweets.text)

# Remove links
tweets.text <- gsub("http\\w+", "", tweets.text)
```

Processing Text with gsub

```
# Remove tabs
tweets.text <- gsub("[ |\t]{2,}", "", tweets.text)

# Remove blank spaces at the beginning
tweets.text <- gsub("^ ", "", tweets.text)

# Remove blank spaces at the end
tweets.text <- gsub(" $", "", tweets.text)
```

Data Preparation NEW SCHOOL

With tm_map

Processing Text with tm_map

```
# Pull down a webpage
u = "http://cran.r-project.org/web/packages/available_packages_by_date.html"

# Read in an HTML Table
t = readHTMLTable(u)[[1]]

# Change to a corpus
ap.corpus <- Corpus(DataframeSource(data.frame(as.character(t[,3]))))

# Remove Punctuation
ap.corpus <- tm_map(ap.corpus, removePunctuation)
```

Processing Text with tm_map

```
#Change to lower case
ap.corpus <- tm_map(ap.corpus, tolower)

#Remove stopwords
ap.corpus <- tm_map(ap.corpus, function(x) removeWords(x, stopwords("english")))

#Add Custom stop words.
ap.corpus <-tm_map(ap.corpus, removeWords, c(stopwords("english"),"my","custom","words"))

## create a term document matrix
dtm <- DocumentTermMatrix(ap.corpus)
```

More Data from the Text - package Sentiment

1. classify_emotion
   This function helps us to analyze some text and classify it in different types of emotion: anger, disgust, fear, joy, sadness, and surprise. The classification can be performed using two algorithms: one is a naive Bayes classifier trained on Carlo Strapparava and Alessandro Valitutti's emotions lexicon; the other one is just a simple voter procedure.

2. classify*polarity*
   *In contrast to the classification of emotions, the classify*polarity function allows us to classify some text as positive or negative.

Kaggle2 - Pizza

1. Clean and prepare text
2. Use Word cloud to identify frequently occurring variables.
3. Code the appearance of specific words to relate to specific constructs.
4. Calculate Sentiment.
5. Visualize relationships
6. Model Relationships

Remove some common terms.

```
c = Corpus(VectorSource(req))
c_clean = tm_map(c, removeWords, c(stopwords('SMART'), 'pizza', 'pizzas', 'request', 'requests'))
c_clean = tm_map(c_clean, rm_space)
```

Combine similar concepts.

```
money = ' money| now| broke| week| until| time| last|day| when| paid| next| first|night|
after| tomorrow| month| while| account| before| long| rent| buy| bank| still| bill| ago| cash| due|
 soon| past| never|check| spent| year| poor| till| morning| dollar| financial| hour| evening| credit| budget|
 loan| buck| deposit| current| pay'
job = ' work| job|check|employ| interview| fire| hire'
student = ' college| student| school| roommate| study| university| final| semester| class|
 project| dorm| tuition'
family = ' family| mom| wife| parent| mother| husband| dad| son| daughter| father| mum'
craving = ' friend| girlfriend| crave| craving| birthday| boyfriend| celebrat| party| parties|
 game| movie| film| date| drunk| beer| invite| drink| waste'
```

Beyond the Basics with APIs

Sentiment analysis (also known as opinion mining) refers to the use of natural language processing, text analysis and computational linguistics to identify and extract subjective information in source materials.

Generally speaking, sentiment analysis aims to determine the attitude of a speaker or a writer with respect to some topic or the overall contextual polarity of a document.

Sentiment Analysis

http://www.alchemyapi.com/api/entity/sentiment.html

Sentiment

http://www.alchemyapi.com/products/demo/

http://www.alchemyapi.com/products/demo/alchemylanguage/

Where to go from Here

Where to go from Here

- Coursera NLP
  https://www.coursera.org/course/nlp
- Text Processing with Python
  http://www.nltk.org/book/
- Text Mining with R
  http://www.jstatsoft.org/v25/i05/paper
- TM_map
  http://cran.r-project.org/web/packages/tm/tm.pdf