Data Computing - Text Data

Chapter 9 from "Data Science and Big Data Analytics: Discovering, Analyzing, Visualizing and Presenting Data"
1st Edition by EMC Education Services

Ka-Chun Wong, Department of Computer Science, City University of Hong Kong
Charles Tappert Seidenberg, School of CSIS, Pace University

Contents

9. Text Analysis

- Text Analysis (or Text Analytics) concerns the representation, processing, and modeling of text data to derive useful insights
- Text Mining is the important component of Text Analysis that discovers the relationships and interesting patterns
- Corpus – large collection of texts (plural of corpus is corpora)
- Dimension – number of distinct words or base forms in corpus
  - The high dimensionality of text is a major issue
  - *Green Eggs and Ham* by Dr. Seuss – 804 total words, 50 different words
- For most of the time, text is not structured

9. Text Analysis
Example Data Sources and Formats for Text Analysis

Data Source	Data Format	Data Structure Type
News articles	TXT, HTML, or Scanned PDF	Unstructured
Literature	TXT, DOC, HTML, or PDF	Unstructured
E-mail	TXT, MSG, or EML	Unstructured
Web pages	HTML	Semi-structured
Server logs	LOG or TXT	Semi-structured or Quasi-structured
Social network API firehoses	XML, JSON, or RSS	Semi-structured
Call center transcripts	TXT	Unstructured

9. Text Analysis

Example Corpora in Natural Language Processing (NLP)

Corpus	Word Count	Domain	Website
Shakespeare	0.88 million	Written	http://shakespeare.mit.edu/
Brown Corpus	1 million	Written	http://icame.uib.no/brown/bcm.html
Penn Treebank	1 million	Newswire	http://www.cis.upenn.edu/~treebank/
Switchboard Phone Conversations	3 million	Spoken	http://catalog.ldc.upenn.edu/LDC97S62
British National Corpus	100 million	Written and spoken	http://www.natcorp.ox.ac.uk/
NA News Corpus	350 million	Newswire	http://catalog.ldc.upenn.edu/LDC95T21
European Parliament Proceedings Parallel Corpus	600 million	Legal	http://www.statmt.org/europarl/
Google N-Grams Corpus	1 trillion	Written	http://catalog.ldc.upenn.edu/LDC2006T13

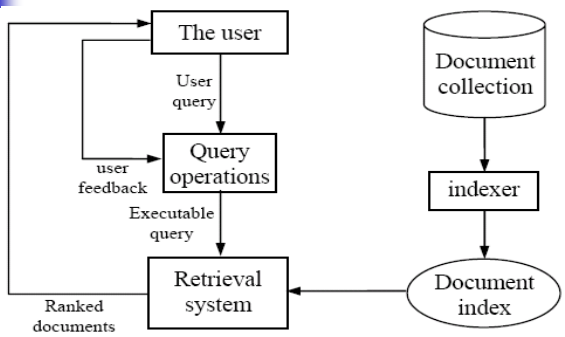
9.1 Text Analysis Steps

- Search and Retrieval
  - Identifies documents in a corpus that contain search items
    - Specific words, phrases, topics, entities – e.g., people, organizations
  - These search items are generally called key terms
- Parsing
  - Imposes a structure on the unstructured text
- Text Mining
  - Discovers meaningful insights in the text
  - Uses techniques such as clustering and classification

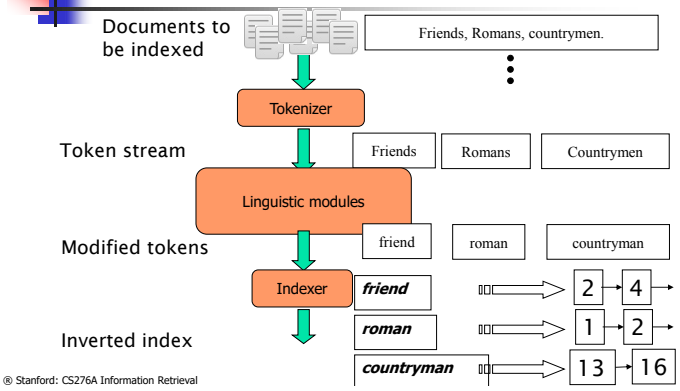
9.1 Text Analysis Steps Search and Retrieval

- Find k objects in the corpus of documents which are most similar to my query.
- Can be viewed as "interactive" data mining - query not specified a priori.
- Main problems of text retrieval:
  - What does "similar" mean?
  - How do I know if I have the right documents?
  - How can I incorporate user feedback?

9.1 Text Analysis Steps Search and Retrieval

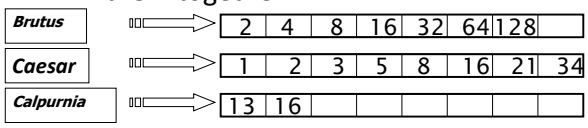


Inverted Index Construction



Query Optimization

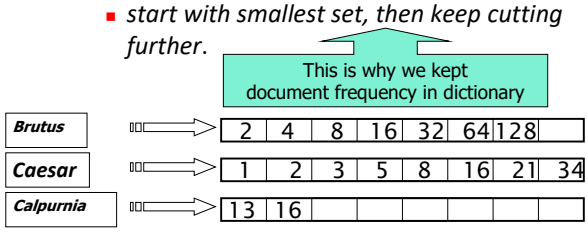
- What is the best order for query processing?
- Consider a query that is an *AND* of *n* terms.
- For each of the *n* terms, get its postings, then *AND* them together.



Query: ***Brutus*** *AND* ***Calpurnia*** *AND* ***Caesar***

Query Optimization Example

- Process in order of increasing frequency:
  - *start with smallest set, then keep cutting further.*



Execute the query as (***Calpurnia*** *AND* ***Brutus)*** *AND* ***Caesar***.

Term Document Matrix

- All Terms (i.e. Words) of Interest (in frequency)
  $<t_1, t_2, t_3, \ldots, t_n>$
- Each Document (in term frequency/weight)
  $D = < a_1, a_2, a_3, \ldots, a_n>$
  a_i = frequency/weight of t_i in D
- Query (in term frequency/weight)
  $Q = < b_1, b_2, b_3, \ldots, b_n>$
  b_i = frequency/weight of t_i in Q
- Text Query = To find a document D_i in D which maximize the similarity (D_i, Q)

® Jian-Yun Nie

Matrix representation

Document space	t_1	t_2	t_3	…	t_n ← Term vector space
D_1	a_{11}	a_{12}	a_{13}	…	a_{1n}
D_2	a_{21}	a_{22}	a_{23}	…	a_{2n}
D_3	a_{31}	a_{32}	a_{33}	…	a_{3n}
…					
D_m	a_{m1}	a_{m2}	a_{m3}	…	a_{mn}
Q	b_1	b_2	b_3	…	b_n

Text Query = To find D_i such that the similarity between D_i and Q is maximized.

Matrix representation example

Term document matrix

Example: 10 documents: 10000 terms

	Database	SQL	Index	Regression	Likelihood	linear
D1	24	21	9	0	0	3
D2	32	10	5	0	3	0
D3	12	16	5	0	0	0
D4	6	7	2	0	0	0
D5	43	31	20	0	3	0
D6	2	0	0	18	7	6
D7	0	0	1	32	12	0
D8	3	0	0	22	4	4
D9	1	0	0	34	27	25
D10	6	0	0	17	4	23

9994 remaining terms

Bag-of-words model

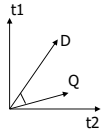
® Jih-Jeng Huang

® Jian-Yun Nie

Example Formulas for Similarity

Dot product $$Sim(D,Q) = \sum_i (a_i * b_i)$$

Cosine $$Sim(D,Q) = \frac{\sum_i (a_i * b_i)}{\sqrt{\sum_i a_i^2 * \sum_i b_i^2}}$$



Dice $$Sim(D,Q) = \frac{2\sum_i (a_i * b_i)}{\sum_i a_i^2 + \sum_i b_i^2}$$

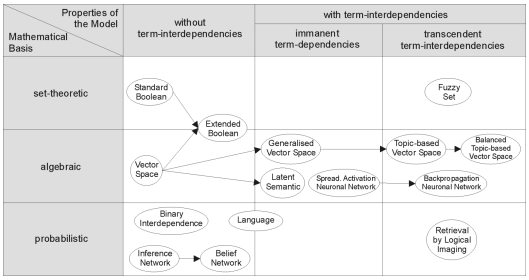
Jaccard $$Sim(D,Q) = \frac{\sum_i (a_i * b_i)}{\sum_i a_i^2 + \sum_i b_i^2 - \sum_i (a_i * b_i)}$$

9.1 Text Analysis Steps Search and Retrieval

- Other Challenges:
  - Phrase Queries
    - We want to be able to answer queries such as ***"stanford university"*** – as a phrase
    - biword indexes
  - Positional Indexes
    - We store the position(s) in which each word occurs in each document

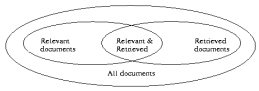
® Stanford: CS276A Information Retrieval

9.1 Text Analysis Steps Search and Retrieval



https://en.wikipedia.org/wiki/Information_retrieval#Model_types

9.1 Text Analysis Steps Search and Retrieval



- Precision: the percentage of retrieved documents that are in fact relevant to the query (i.e., "correct" responses)

$$precision = \frac{|\{Relevant\} \cap \{Retrieved\}|}{|\{Retrieved\}|}$$

- Recall: the percentage of documents that are relevant to the query and were, in fact, retrieved

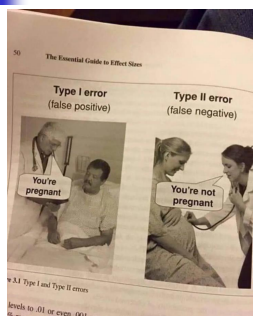
$$recall = \frac{|\{Relevant\} \cap \{Retrieved\}|}{|\{Relevant\}|}$$

9.1 Text Analysis Steps Search and Retrieval

	Truth:Relvant	Truth:Not Relevant
Algorithm:Relevant	TP	FP
Algorithm: Not Relevant	FN	TN

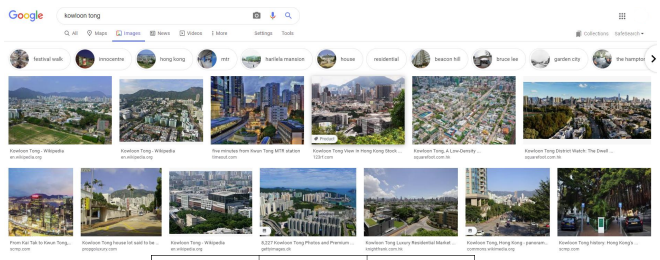
- We've been here before!
  - Precision = TP/(TP+FP)
  - Recall = TP/(TP+FN)
  - Trade off:
    - If algorithm is 'picky': precision high, recall low
    - If algorithm is 'relaxed': precision low, recall high

9.1 Text Analysis Steps Search and Retrieval



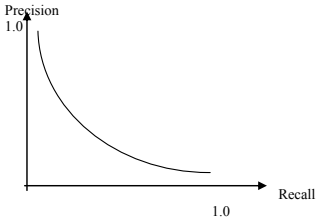
	Truth:Relvant	Truth:Not Relevant
Algorithm:Relevant	TP	FP
Algorithm: Not Relevant	FN	TN

9.1 Text Analysis Steps Search and Retrieval



	Truth:Relvant	Truth:Not Relevant
Algorithm:Relevant	TP	FP
Algorithm: Not Relevant	FN	TN

General form of precision/recall



-Precision change w.r.t. Recall (not a fixed point)

-Systems cannot be compared at one Precision/Recall point

-Average precision (on 11 points of recall: 0.0, 0.1, …, 1.0)

9.1 Text Analysis Steps

- Search and Retrieval
  - Identifies documents in a corpus that contain search items
    - Specific words, phrases, topics, entities – e.g., people, organizations
  - These search items are generally called key terms
- Parsing (also known as NLP)
  - Imposes a structure on the unstructured text
- Text Mining
  - Discovers meaningful insights in the text
  - Uses techniques such as clustering and classification

9.1 Text Analysis Steps Parsing

- Part-of-Speech (POS) Tagging
  - "he saw a fox" => PRP VBD DT NN
    - pronoun (PRP), verb (VBD), determiner (DT), noun (NN)
- Lemmatization finds dictionary base forms
  - obesity causes many problems =>
  - obesity cause many problem
- Stemming (e.g., Porter's stemming algorithm)
  - Similar to lemmatization but dictionary not required
  - obesity causes many problems => obes caus mani problem

import nltk

9.1 Text Analysis Steps
Lexical analysis (part-of-speech tagging)

This	sentence	serves	as	an	example	of	annotated	text…
Det	N	V1	P	Det	N	P	V2	N

"This is a new sentence." → POS Tagger → This is a new sentence.
Det Aux Det Adj N

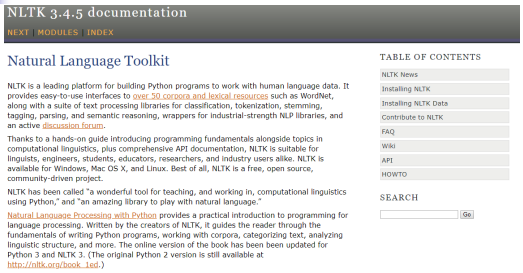
Pick the most likely tag sequence.

$$p(w_1,\ldots,w_k,t_1,\ldots,t_k) = \begin{cases} p(t_1 \mid w_1)\ldots p(t_k \mid w_k)\, p(w_1)\ldots p(w_k) \\ \prod_{i=1}^{k} p(w_i \mid t_i)\, p(t_i \mid t_{i-1}) \end{cases}$$

Independent assignment
Most common tag

Partial dependency
(Hidden Markov Model)

Python NLTK



https://www.nltk.org/book/ch05.html

9.1 Text Analysis Steps
Lexical analysis (part-of-speech tagging)

```
>>> import nltk
>>> text = word_tokenize("And now for something completely different")
>>> nltk.pos_tag(text)
[('And', 'CC'), ('now', 'RB'), ('for', 'IN'), ('something', 'NN'),
('completely', 'RB'), ('different', 'JJ')]
```

- CC coordinating conjunction
- CD cardinal digit
- DT determiner
- EX existential there (like: "there is" … think of it like "there exists")
- FW foreign word
- IN preposition/subordinating conjunction
- JJ adjective 'big'
- JJR adjective, comparative 'bigger'
- JJS adjective, superlative 'biggest'
- LS list marker 1)
- MD modal could, will
- NN noun, singular 'desk'
- NNS noun plural 'desks'
- NNP proper noun, singular 'Harrison'
- NNPS proper noun, plural 'Americans'
- PDT predeterminer 'all the kids'
- POS possessive ending parent's
- PRP personal pronoun I, he, she
- PRP$ possessive pronoun my, his, hers
- RB adverb very, silently,
- RBR adverb, comparative better
- RBS adverb, superlative best
- RP particle give up
- TO, to go 'to' the store.
- UH interjection, errrrrrrm
- VB verb, base form take
- VBD verb, past tense took
- VBG verb, gerund/present participle taking
- VBN verb, past participle taken
- VBP verb, sing. present, non-3d take
- VBZ verb, 3rd person sing. present takes
- WDT wh-determiner which
- WP wh-pronoun who, what
- WP$ possessive wh-pronoun whose
- WRB wh-adverb where, when

9.1 Text Analysis Steps
Lemmatization

Lemmatization

- **(With Dictionary)**
- Reduce inflectional/variant forms to base form
- E.g.,
  - *am, are, is* → *be*
  - *car, cars, car's, cars'* → *car*
- *the boy's cars are different colors* → *the boy car be different color*

9.1 Text Analysis Steps Lemmatization

Lemmatization

```
# import these modules
from nltk.stem import WordNetLemmatizer

lemmatizer = WordNetLemmatizer()

print("rocks :", lemmatizer.lemmatize("rocks"))
print("corpora :", lemmatizer.lemmatize("corpora"))

# a denotes adjective in "pos"
print("better :", lemmatizer.lemmatize("better", pos ="a"))
```

Output :

```
rocks : rock
corpora : corpus
better : good
```

https://www.geeksforgeeks.org/python-lemmatization-with-nltk/

9.1 Text Analysis Steps Stemming

Stemming

e.g. Porter algorithm
(Porter, M.F., 1980, An algorithm for suffix stripping, *Program*, **14**(3) :130-137)

- **(Without Dictionary)**
- Reduce terms to their "roots" before indexing
  - language dependent
  - e.g., ***automate(s), automatic, automation*** all reduced to ***automat***.

for example compressed and compression are both accepted as equivalent to compress.

for exampl compres and compres are both accept as equival to compres.

9.1 Text Analysis Steps Stemming

Stemming

```
# importing modules
from nltk.stem import PorterStemmer
from nltk.tokenize import word_tokenize

ps = PorterStemmer()

sentence = "Programers program with programing languages"
words = word_tokenize(sentence)

for w in words:
    print(w, " : ", ps.stem(w))
```

Output :

```
Programers  :  program
program  :  program
with  :  with
programing  :  program
languages  :  languag
```

https://www.geeksforgeeks.org/python-stemming-words-with-nltk/

9.1 Text Analysis Steps Parsing

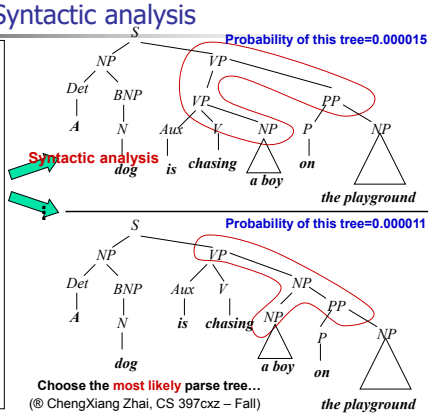
Language-specificity

- Many of the above parsing procedures embody transformations that are
  - Language-specific and
  - Often, application-specific
- These are "plug-in" software libraries to the indexing process
- Both open source and commercial plug-ins available for handling these

9.1 Text Analysis Steps
Syntactic analysis

Probabilistic CFG

Grammar	$S \rightarrow NP\ VP$	1.0
	$NP \rightarrow Det\ BNP$	0.3
	$NP \rightarrow BNP$	0.4
	$NP \rightarrow NP\ PP$	0.3
	$BNP \rightarrow N$	…
	$VP \rightarrow V$	
	$VP \rightarrow Aux\ V\ NP$	…
	$VP \rightarrow VP\ PP$	…
	$PP \rightarrow P\ NP$	1.0
Lexicon	$V \rightarrow chasing$	0.01
	$Aux \rightarrow is$	
	$N \rightarrow dog$	0.003
	$N \rightarrow boy$	
	$N \rightarrow playground$	…
	$Det \rightarrow the$	
	$Det \rightarrow a$	…
	$P \rightarrow on$	



Choose the most likely parse tree…

http://www.nltk.org/book/ch08.html

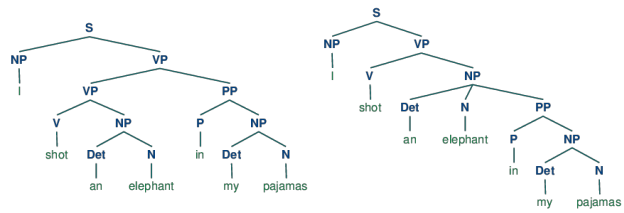
9.1 Text Analysis Steps
Syntactic analysis

```
>>> groucho_grammar = nltk.CFG.fromstring("""
... S -> NP VP
... PP -> P NP
... NP -> Det N | Det N PP | 'I'
... VP -> V NP | VP PP
... Det -> 'an' | 'my'
... N -> 'elephant' | 'pajamas'
... V -> 'shot'
... P -> 'in'
... """)
```

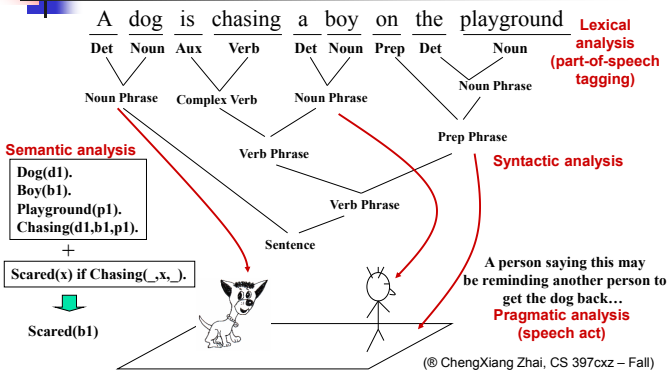
```
>>> sent = ['I', 'shot', 'an', 'elephant', 'in', 'my', 'pajamas']
>>> parser = nltk.ChartParser(groucho_grammar)
>>> for tree in parser.parse(sent):
...     print(tree)
...
(S
  (NP I)
  (VP
    (VP (V shot) (NP (Det an) (N elephant)))
    (PP (P in) (NP (Det my) (N pajamas)))))
(S
  (NP I)
  (VP
    (V shot)
    (NP (Det an) (N elephant) (PP (P in) (NP (Det my) (N pajamas))))))
```

http://www.nltk.org/book/ch08.html

9.1 Text Analysis Steps
Syntactic analysis



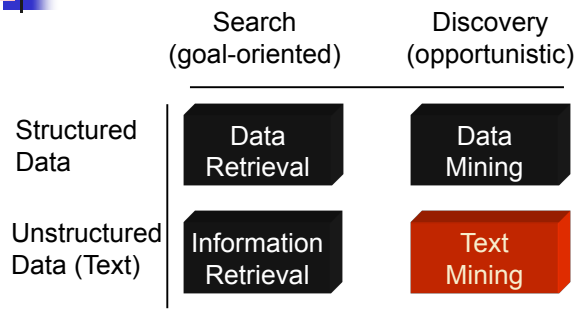
9.1 Text Analysis Steps
Parsing



9.1 Text Analysis Steps

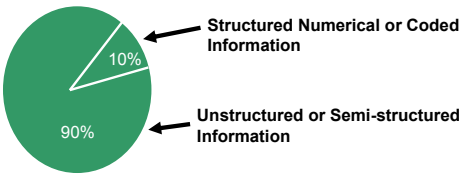
- Search and Retrieval
  - Identifies documents in a corpus that contain search items
    - Specific words, phrases, topics, entities – e.g., people, organizations
  - These search items are generally called key terms
- Parsing
  - Imposes a structure on the unstructured text
- Text Mining
  - Discovers meaningful insights in the text
  - Uses techniques such as clustering and classification

9.1 Text Analysis Steps Text Mining



9.1 Text Analysis Steps Text Mining

- Approximately **90%** of the world's data is held in unstructured formats (source: Oracle Corporation)
- Information intensive business processes demand that we transcend from simple document retrieval to "knowledge" discovery.



9.1 Text Analysis Steps Text Mining

	Data Mining	Text Mining
Data Object	Numerical & categorical data	Textual data
Data Structure	Structured	Unstructured & Semi-structured
Data Representation	Straightforward	Complex
Space Dimension	<1000	> 10,000
Maturity	Since 1994	Since 2000
Market	10^5 analysts at large and mid size companies	10^8 analysts corporate workers and individual users

9.1 Text Analysis Steps Text Mining

Data Mining
- Identify data sets
- Select features
- Prepare data
- Analyze distribution

Text Mining
- Identify documents
- **Extract features**
- **Select features by text algorithms**
- Prepare data
- Analyze distribution

® "Text Mining: Finding Nuggets in Mountains of Textual Data" by Jochen Dorre, Peter Gerstl, and Roland Seiffert

9.1 Text Analysis Steps Text Mining

Term document matrix

Example: 10 documents: 10000 terms

	Database	SQL	Index	Regression	Likelihood	linear
D1	24	21	9	0	0	3
D2	32	10	5	0	3	0
D3	12	16	5	0	0	0
D4	6	7	2	0	0	0
D5	43	31	20	0	3	0
D6	2	0	0	18	7	6
D7	0	0	1	32	12	0
D8	3	0	0	22	4	4
D9	1	0	0	34	27	25
D10	6	0	0	17	4	23

9994 remaining terms .....

Bag-of-words model

® Jih-Jeng Huang

9.1 Text Analysis Steps Text Mining

- Feature Selection/Building Methods
  - Traditional Methods in Data Mining
    - Principal Component Analysis (PCA)
    - Non-negative Matrix Factorization (NMF)
    - Supervised Learning - Wrapper Approach
    - Information Theory
  - Latent Semantic Analysis (LSA)
  - Latent Dirichlet Allocation (LDA)
  - Word Embedding (e.g. Word2vec)

9.1 Text Analysis Steps Text Mining

- Similarities between Two Documents
  - Traditional Metrics in Data Mining
    - Euclidean Distance
    - Hamming Distance
  - Jaccard Index
  - Cosine Similarity

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

https://en.wikipedia.org/wiki/Cosine_similarity

9.1 Text Analysis Steps Text Mining

- Keyword-based association analysis
- Automatic document classification
- Similarity detection
  - Cluster documents by a common author
  - Cluster documents containing information from a common source
- Link analysis: unusual correlation between entities
- Sequence analysis: predicting a recurring event
- Anomaly detection: find information that violates usual patterns
- Hypertext analysis
  - Patterns in anchors/links
    - Anchor text correlations with linked objects

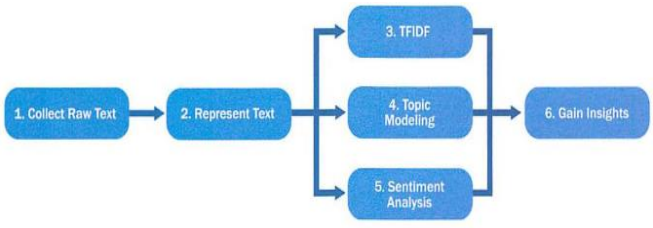
Lecture Break

9.2 Text Analysis Example

- Fictitious company ACME
  - Makes two products – *bPhone* and *bEbook*
- ACME monitors social media and popular review sites
  - Are people mentioning its products?
  - What is being said?
  - Are the products seen as good or bad?
  - If people say ACME product is bad, why?
    - For example, are they complaining about battery life of the bPhone or response time in their bEbook?

9.2 Text Analysis Example

- ACME's Text Analysis Process



9.3 Collecting Raw Data

- The text data must first be collected
- ACME is interested in what the reviews say about *bPhone* and *bEbook* and when the reviews are posted
- Many websites and services offer public APIs for third-party developers to access their data
  - For example, Twitter APIs can retrieve public Twitter posts that contain the specified keywords.

https://developer.twitter.com/en/docs.html

9.3 Collecting Raw Data

- Use web scraper to extract useful web info
- Use the curl tool to fetch HTML source code given specific URLs
- Use Xpath and regular expressions to select and extract data that matches certain patterns
- Regular expressions can find words and strings that match particular patterns of interest
- Commercial software are also available.

https://www.crummy.com/software/BeautifulSoup/bs4/doc/

9.3 Collecting Raw Data
Example Regular Expressions

Regular Expression	Matches	Note
`b(P|p)hone`	bPhone, bphone	Pipe "|" means "or"
`bEbo*k`	bEbk, bEbok, bEbook, bEboook, bEbooook, bEboooook, …	"*" matches zero or more occurrences of the preceding letter
`bEbo+k`	bEbok, bEbook, bEboook, bEbooook, bEboooook, …	"+" matches one or more occurrences of the preceding letter
`bEbo{2,4}k`	bEbook, bEboook, bEbooook	"{2,4}" matches from two to four repetitions of the preceding lette "o"
`^I love`	Text starting with "I love"	"^" matches the start of a string
`ACME$`	Text ending with "ACME"	"$" matches the end of a string

9.4 Representing Text

- Tokenization – separates words from the text

```
I once had a gf back in the day. Then the bPhone came out
```

tokenization based on spaces would output a list of tokens.

```
{I, once, had, a, gf, back, in, the, day.,
Then, the, bPhone, came, out, lol}
```

- Case folding – reduces all letters to lowercase

```
i once had a gf back in the day. then the bphone came out lol
```

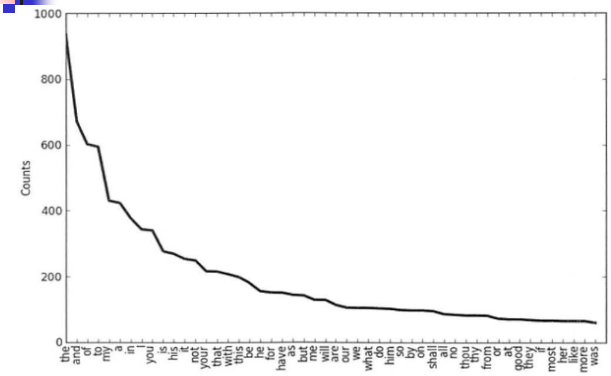
  - Problems – e.g., WHO = World Health Organization

9.4 Representing Text

- Term frequency (TF) – easily calculated from bag-of-words representation
  - See figure next slide
  - Roughly follows Zipf's Law – the frequency of a word is inversely proportional to its rank in the frequency table

9.4 Representing Text

50 most frequent words in Shakespeare's Hamlet



9.5 Term Frequency – Inverse Document Frequency (TFIDF)

- Term Frequency (TF) is a widely used measure in text analysis: robust and efficient.

$$TF_1(t,d) = \sum_{i=1}^{n} f(t,t_i) \qquad t_i \in d; |d| = n$$

$$f(t,t') = \begin{cases} 1, & \text{if } t = t' \\ 0, & \text{otherwise} \end{cases}$$

9.5 Term Frequency – Inverse Document Frequency (TFIDF)

- Term frequency highlights common words
  - Eliminate stop words, such as *the, a, of, and*
- To fix this problem, consider the metrics
  - Document frequency (DF) = the number of documents in the corpus that contain the term

$$DF(t) = \sum_{i=1}^{N} f'(t,d_i) \qquad d_i \in D; |D| = N$$

$$D = \{d_1, d_2, \ldots d_N\}$$

$$f'(t,d') = \begin{cases} 1, & \text{if } t \in d' \\ 0, & \text{otherwise} \end{cases}$$

9.5 Term Frequency – Inverse Document Frequency (TFIDF)

- Inverted document frequency (IDF) is obtained by dividing N (the total number of documents).
- In log form as

$$IDF_1(t) = \log \frac{N}{DF(t)}$$

- Or to avoid division-by-zero as

$$IDF_2(t) = \log \frac{N}{DF(t)+1}$$

9.5 Term Frequency – Inverse Document Frequency (TFIDF)

- Words with high IDF tend to be more meaningful over the entire corpus
- There is still a problem with IDF
  - Because the document count in a corpus (N) remains constant, IDF depends solely on DF

9.5 Term Frequency – Inverse Document Frequency (TFIDF)

- TFIDF (or TF-IDF) involves both TF and IDF

$$TFIDF(t,d) = TF(t,d) \times IDF(t)$$

  - TFIDF scores words higher that appear more often in a document but less often across all documents
  - TFIDF applies to a term in a specific document, so it gets different scores in different documents
  - Reveals little of inter- or intra-document structure

9.5 Term Frequency – Inverse Document Frequency (TFIDF)

- Given a document containing terms with given frequencies:
  Kent = 4; Ohio = 5; University = 10
  and assume a collection of 10,000 documents and document frequencies of these terms are:
  Kent = 1000; Ohio = 100; University = 5000.

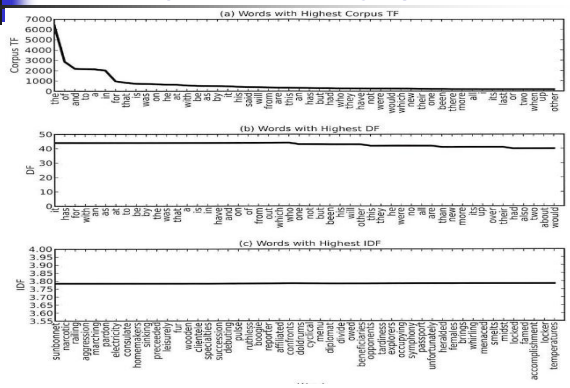
THEN

Kent:	TF = 4; IDF = log(10000/1000) = 1; TFIDF = 4
Ohio:	TF = 5; IDF = log(10000/100) = 2; TFIDF = 10
University:	TF = 10; IDF = log(10000/5000) = 0.3; TFIDF = 3

Modified from Ch4. of Hakam Alomari

9.5 Term Frequency

Brown Corpus news category: TF, DF, IDF



9.6 Categorizing Documents by Topics

- Returning to the ACME example, the team wants to categorize the reviews by topic
- Topic modeling – prevalent statistical approach
  - Uncovers hidden topical patterns within a corpus
  - Annotates documents according to these topics
  - Uses annotations to organize, search, and summarize texts
- A topic is formally defined as a distribution over a fixed vocabulary of words

9.6 Categorizing Documents by Topics

- Latent Dirichlet Allocation (LDA) topic model
  - Simple generative probabilistic model of a corpus
  - Data treated as a result of a generative process that includes hidden variables
  - Assumes fixed vocabulary of words
  - Assumes constant predefined number of topics

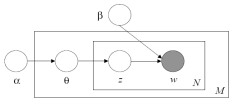
9.6 Categorizing Documents by Topics

Latent Dirichlet Allocation (LDA)

α=per-document topic distributions

Assume data was generated by a generative process:
θ is a probability distribution of topics (a document sampled from α)
z is a probability distribution of words (a topic sampled from θ)
w is a word, the only real observable (a word sampled from z)
(N=number of words in a document; M=number of documents)

1. Choose $N \sim Poisson(\xi)$
2. Choose $\theta \sim Dir(\alpha)$
3. For each of the N words w_n:
   (a) Choose a topic $z_n \sim Multinomial(\theta)$
   (b) Choose a word w_n from $p(w_n|z_n,\beta)$, a multinomial probability conditioned on the topic z_n

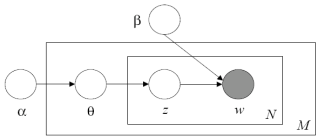


9.6 Categorizing Documents by Topics
Latent Dirichlet Allocation (LDA)

$$p(\theta, \mathbf{z}, \mathbf{w}|\alpha, \beta) = p(\theta|\alpha)\prod_{n=1}^{N} p(z_n|\theta)p(w_n|z_n, \beta)$$

$$p(\mathbf{w}|\alpha, \beta) = \int p(\theta|\alpha)\left(\prod_{n=1}^{N}\sum_{z_n} p(z_n|\theta)p(w_n|z_n, \beta)\right)d\theta$$

$$p(D|\alpha, \beta) = \prod_{d=1}^{M}\int p(\theta_d|\alpha)\left(\prod_{n=1}^{N_d}\sum_{z_{dn}} p(z_{dn}|\theta_d)p(w_{dn}|z_{dn}, \beta)\right)d\theta_d$$



For details, please see Blei, et al 2003
http://www.jmlr.org/papers/volume3/blei03a/blei03a.pdf

9.6 Categorizing Documents by Topics
Latent Dirichlet Allocation (LDA)

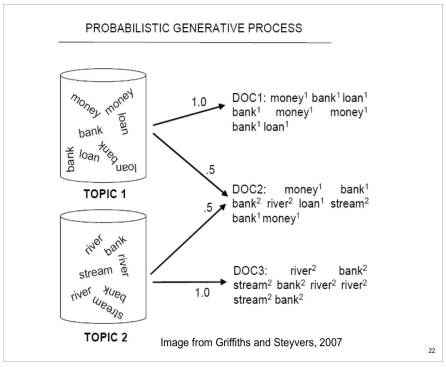
In Python,

```
from sklearn.feature_extraction.text import CountVectorizer

# the vectorizer object will be used to transform text to vector
form
vectorizer = CountVectorizer(max df=0.9, min df=25, token pattern
='\w+|\$[\d\.]+|\S+')

# apply transformation
tf = vectorizer.fit_transform(df['clean_tweet']).toarray()

# tf feature names tells us what word each column in the matric r
epresents
tf_feature_names = vectorizer.get_feature_names()
```

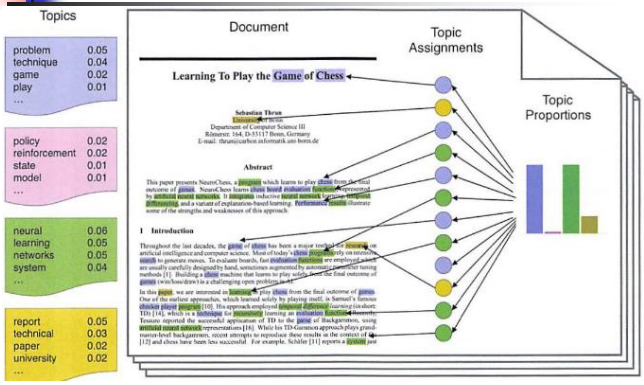
```
from sklearn.decomposition import LatentDirichletAllocation

number_of_topics = 10

model = LatentDirichletAllocation(n components=number of topics,
random_state=0)

model.fit(tf)
```

https://ourcodingclub.github.io/tutorials/topic-modelling-python/

9.6 Categorizing Documents by Topics
Figure illustrating intuitions behind LDA



9.6 Categorizing Documents by Topics
Figure illustrating intuitions behind LDA



9.7 Determining Sentiments

- Sentiment analysis is a group of tasks that use statistics and NLP to mine opinions from texts
- Make lists of positive and negative words
  - Positive – brilliant, awesome, spectacular
  - Negative – awful, stupid, hideous
  - This simple approach achieves about 60% accuracy
- Naïve Bayes, maximum entropy, and SVM
  - Can achieve about 80% accuracy

9.7 Determining Sentiments

In Python,

```
>>> from textblob import TextBlob

>>> testimonial = TextBlob("Textblob is amazingly simple to use. What great fun!")

>>> testimonial.sentiment

Sentiment(polarity=0.39166666666666666,subjectivity=0.4357142857142857)

>>> testimonial.sentiment.polarity

0.39166666666666666
```

https://textblob.readthedocs.io/en/dev/quickstart.html

9.7 Determining Sentiments

- Emoticons can make it easy and fast to detect sentiment but this method can be misleading
- E.g., the text below with :) emoticon does not necessarily correspond to a positive sentiment

9.8 Gaining Insights

- Returning to the ACME example used in this chapter, this section shows how various techniques can be used to gain insights into customer opinions
  - For simplicity, only the *bPhone* product is used here
- The ACME data science team collects 300 reviews
  - Using the keyword *bPhone*
- After tokenization, removing stop words, and case folding to lowercase, the 300 reviews are visualized as a word cloud with more frequently appearing words in larger font size

9.8 Gaining Insights

In Python,

```
import os

from os import path
from wordcloud import WordCloud

# get data directory (using getcwd() is needed to support running example in generated
IPython notebook)
d = path.dirname(__file__) if "__file__" in locals() else os.getcwd()

# Read the whole text.
text = open(path.join(d, 'constitution.txt')).read()

# Generate a word cloud image
wordcloud = WordCloud().generate(text)

# Display the generated image:
# the matplotlib way:
import matplotlib.pyplot as plt
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
```

https://amueller.github.io/word_cloud/auto_examples/simple.html

9.8 Gaining Insights

Word cloud on all 300 *bPhone* reviews

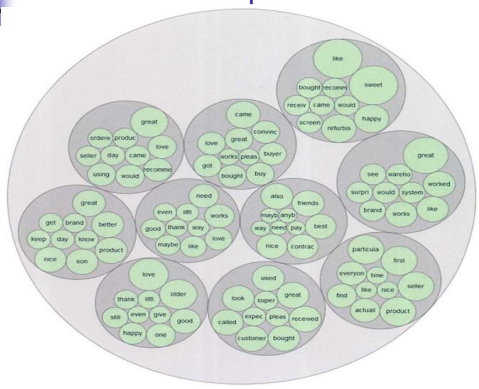
Often remove domain-specific stop words not useful for the study. In this case, remove word like *phone, bPhone,* and *ACME*.

9.8 Gaining Insights

Word cloud on five-stars reviews

9.8 Gaining Insights

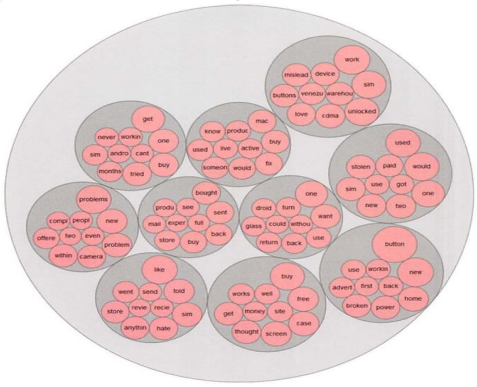
LDA model: ten topics on five-star reviews



9.8 Gaining Insights
Word cloud on one-star reviews

Note the words *sim, button, stolen, venezuela*. Further investigation revealed unauthorized sellers in Venezuela sold stolen *bPhones*.

9.8 Gaining Insights
LDA model: ten topics on one-star reviews



9.8 Gaining Insights
Reviews highlighted by TFIDF values

minor bugs September 17, 2013
this was for my sister who loves it. she says it has minor bugs but nothing she cant deal with. she is overall satisfied with it

mint condition ! ! :) September 13, 2013
great price . not a scratch or bump on the bphone ! it came a lot speedier than expected so thats always a plus ! its just wonderful . only had it for a couple of days and could n't ask for anything more ! ! ! !

buttons did not work September 08, 2013
when i went to have my contacts transferred it was found that the two buttons need to switch did not work consistantly

it 's a bphone. August 12, 2013
i hate acme and acme products. base both on principle and on functionality (or lack thereof) . that being said i guess this phone is great for old people that are n't tech savvy. i bought this for my aunt .

just like new i love this phone August 10, 2013
my phone is great it came on time with everything in the box and new. nothing was use . it was all in its original packaging .

great product August 10, 2013
i lost my phone on halloween and needed a quick and easy replacement . the phone was in very good condition . and had met my expectation .

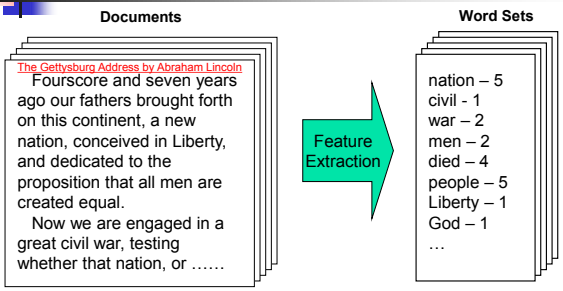
bphone July 26, 2013
the phone was clean . unscratched and in good condition. i just wish the battery lasted a little longer. otherwise a great purchase !

bphone July 25, 2013
love my bphone i have not had any problems with it. it 's easy to use and holds so much more than my last phone .

9.8 Gaining Insights
Sentiment analysis on over 100 tweets

Indicates most customers satisfied with ACME's bPhone.

Limitation



Loses all order-specific information!
Severely limits context!

https://kleiber.me/blog/2018/02/25/top-10-python-nlp-libraries-2018/

Python NLP packages

Library	Outstanding Function/Feature	GitHub Stars
spaCy	Extremely optimized NLP library that is meant to be operated together with deep learning frameworks such as *TensorFlow* or *PyTorch*.	8343
Gensim	Highly efficient and scalable topic/semantic modelling.	6312
Pattern	Web (data) mining / crawling and common NLP tasks.	6095
NLTK	The 'mother' of all NLP libraries. Excellent for educational purposes and the de-facto standard for many NLP tasks.	6022
TextBlob	Modern multi-purpose NLT toolset that is really great for fast and easy development.	4807
Polyglot	Multilingualism and transliteration capabilities.	754
Vocabulary	Retrieve semantic information from individual words.	397
PyNLPl	Extensive functionality regarding FoLiA XML and many other common NLP format (CQL, Giza, Moses, ARPA, Timbl, etc.).	294
Stanford CoreNLP Python	Reliable, robust and accurate NLP platform based on a client-server architecture. Written in Java, and accessible trough multiple Python wrapper libraries.	188
MontyLingua	End-to-end NLP processor working with Python and Java. Historical!	-

spaCy https://spacy.io/

- Support for **64+ languages**
- **63 trained pipelines** for 19 languages
- Multi-task learning with pretrained **transformers** like BERT
- Pretrained **word vectors**
- State-of-the-art speed
- Production-ready **training system**
- Linguistically-motivated **tokenization**
- Components for **named entity** recognition, part-of-speech tagging, dependency parsing, sentence segmentation, **text classification**, lemmatization, morphological analysis, entity linking and more
- Easily extensible with **custom components** and attributes
- Support for custom models in **PyTorch**, **TensorFlow** and other frameworks
- Built in **visualizers** for syntax and NER
- Easy **model packaging**, deployment and workflow management
- Robust, rigorously evaluated accuracy

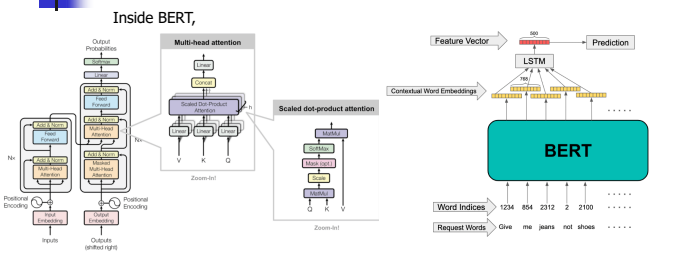
spaCy https://spacy.io/

```
# pip install -U spacy
# python -m spacy download en_core_web_sm
import spacy

# Load English tokenizer, tagger, parser and NER
nlp = spacy.load("en_core_web_sm")

# Process whole documents
text = ("When Sebastian Thrun started working on sel
        "Google in 2007, few people outside of the c
        "seriously. “I can tell you very senior CEOs
        "car companies would shake my hand and turn
        "worth talking to,” said Thrun, in an interv
        "this week.")
doc = nlp(text)

# Analyze syntax
print("Noun phrases:", [chunk.text for chunk in doc.noun_chunks])
print("Verbs:", [token.lemma_ for token in doc if token.pos_ == "VERB"])

# Find named entities, phrases and concepts
for entity in doc.ents:
    print(entity.text, entity.label_)
```

RUN

```
Noun phrases: ['Sebastian Thrun', 'self-driving cars', 'Google', 'few people',
'the company', 'him', 'I', 'you', 'very senior CEOs', 'major American car compan
ies', 'my hand', 'I', 'Thrun', 'an interview', 'Recode']
Verbs: ['start', 'work', 'drive', 'take', 'tell', 'shake', 'turn', 'be', 'talk',
'say']
Sebastian Thrun PERSON
2007 DATE
American NORP
Thrun PERSON
Recode PERSON
earlier this week DATE
```

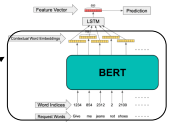
BERT (deep learning)



https://neptune.ai/blog/bert-and-the-transformer-architecture https://multithreaded.stitchfix.com/blog/2019/07/15/give-me-jeans/

Use of BERT (deep learning)

```
import torch
from transformers import BertTokenizer, BertModel

tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')
model = BertModel.from_pretrained('bert-base-uncased',output_hidden_states = True)
model.eval()

text = "After stealing money from the bank vault, the bank robber was seen"
# Add the special tokens.
marked_text = "[CLS] " + text + " [SEP]"
tokenized_text = tokenizer.tokenize(marked_text)
indexed_tokens = tokenizer.convert_tokens_to_ids(tokenized_text)
segments_ids = [1] * len(tokenized_text)

# Convert inputs to PyTorch tensors
tokens_tensor = torch.tensor([indexed_tokens])
segments_tensors = torch.tensor([segments_ids])

with torch.no_grad():
    outputs = model(tokens_tensor, segments_tensors)
```



Complete Google Code:
https://colab.research.google.com/drive/1yFphU6PW9Uo6lmDly_ud9a6c4RCYlwdX#scrollTo=Pg0P9rFxJwwp

LLMs (Large Language Models)

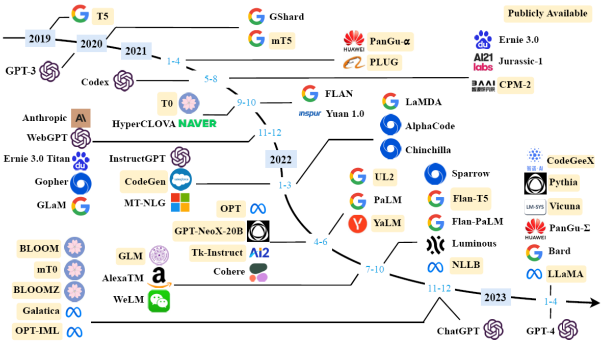


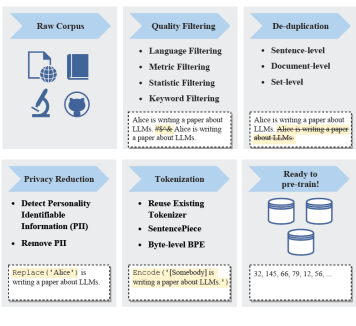
Fig. 1. A timeline of existing large language models (having a size larger than 10B) in recent years. The timeline was established mainly according to the release date (e.g., the submission date to arXiv) of the technical paper for a model. If there was not a corresponding paper, we set the date of a model as the earliest time of its public release or announcement. We mark the LLMs with publicly available model checkpoints in yellow color. Due to the space limit of the figure, we only include the LLMs with publicly reported evaluation results.

https://arxiv.org/pdf/2303.18223.pdf

LLMs (Large Language Models)
Typical LLM Building Steps

TABLE 2
Statistics of commonly-used data sources.

Corpora	Size	Source	Latest Update Time
BookCorpus [122]	5GB	Books	Dec-2015
Gutenberg [123]	-	Books	Dec-2021
C4 [73]	800GB	CommonCrawl	Apr-2019
CC-Stories-R [124]	31GB	CommonCrawl	Sep-2019
CC-NEWS [27]	78GB	CommonCrawl	Feb-2019
REALNEWS [125]	120GB	CommonCrawl	Apr-2019
OpenWebText [126]	38GB	Reddit links	Mar-2023
Pushift.io [127]	2TB	Reddit links	Mar-2023
Wikipedia [128]	21GB	Wikipedia	Mar-2023
BigQuery [129]	-	Codes	Mar-2023
the Pile [130]	800GB	Other	Dec-2020
ROOTS [131]	1.6TB	Other	Jun-2022



https://arxiv.org/pdf/2303.18223.pdf

Summary

- 9.1 Text Analysis Steps
- 9.2 A Text Analysis Example
- 9.3 Collecting Raw Text
- 9.4 Representing Text
- 9.5 Term Frequency – Inverse Document Frequency
- 9.6 Categorizing Documents by Topics
- 9.7 Determining Sentiments
- 9.8 Gaining Insights
- Python NLP packages

Reference

Text Analysis Video

- http://www.maxqda.com/excellent-video-introduction-to-the-principles-of-text-analysis-by-prof-lance-gravlee (7 min)

9.1 Text Analysis Steps

- Parsing
  - Imposes a structure on the unstructured text
- Search and Retrieval (e.g. duckduckgo.com)
  - Identifies documents in a corpus that contain search items
    - Specific words, phrases, topics, entities – e.g., people, organizations
  - These search items are generally called key terms
- Text Mining
  - Discovers meaningful insights in the text
  - Uses techniques such as clustering and classification

9.4 Representing Text

- Bag-of-words – represents text as set of terms
  - Widely-used but naïve approach that eliminates order
    - "a dog bites a man" is equivalent to "a man bites a dog"
  - Still considered a good approach

In reviewing the extensive literature accumulated during the past 25 years in the area of retrieval system evaluation, the overwhelming evidence is that the judicious use of single-term identifiers is preferable to the incorporation of more complex entities extracted from the texts themselves or obtained from available vocabulary schedules.

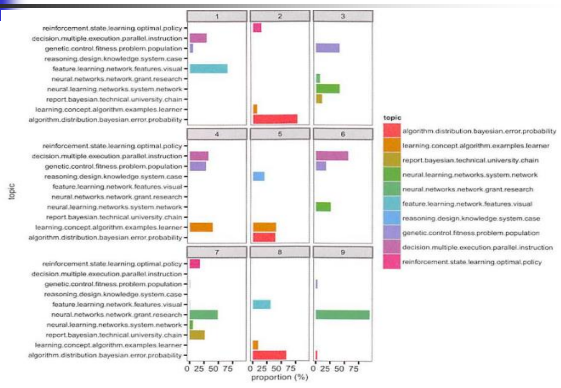
9.4 Representing Text

- Morphological features – additional info such a POS tag, named entities, etc.
  - The features are usually designed for a specific task
  - Creating the features can be a text analysis task in itself
  - One such example is topic modeling, a method to quickly analyze large volumes of text to identify the topic
- Information content (IC) – a metric to denote the importance of a term in a corpus
  - The next section, 9.5, discusses such a metric

9.4 Representing Text
Categories of the Brown Corpus

Category	Number of Sources	Example Source
A. Reportage	44	*Chicago Tribune*
B. Editorial	27	*Christian Science Monitor*
C. Reviews	17	*Life*
D. Religion	17	*William Pollard: Physicist and Christian*
E. Skills and Hobbies	36	*Joseph E. Choate: The American Boating Scene*
F. Popular Lore	48	*David Boroff: Jewish Teen-Age Culture*
G. Belles Lettres, Biography, Memoirs, and so on	75	*Selma J. Cohen: Avant-Garde Choreography*
H. Miscellaneous	30	*U. S. Dep't of Defense: Medicine in National Defense*
J. Learned	80	*J. F. Vedder: Micrometeorites*
K. General Fiction	29	*David Stacton: The Judges of the Secret Court*

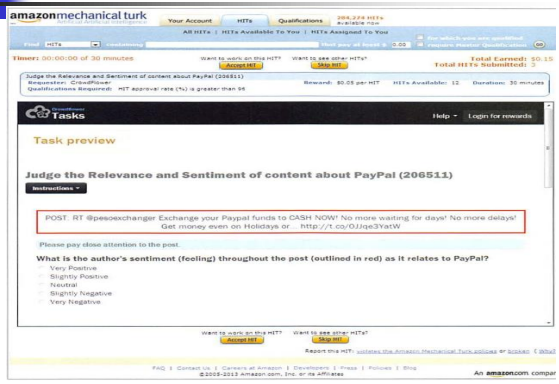
9.6 Categorizing Documents by Topics
Distribution of ten topics over nine documents



9.7 Determining Sentiments
Amazon Mechanical Turk (MTurk)

- To address problems mentioned above, Amazon Mechanical Turk (MTurk) can be used
- It is a crowdsourcing Internet marketplace that enables individuals or businesses to coordinate the use of human intelligence to perform tasks difficult for computers
- MTurk performs Human Intelligence Tasks (HITs)
  - For example, for the tweets illustrative example, human workers can be asked to tag each tweet as positive, neutral, or negative

9.7 Determining Sentiments
Amazon Mechanical Turk (MTurk)



9.7 Determining Sentiments
Evaluation of prediction models

- Data usually split into training and testing sets
- Supervised learning – labeled data
- Confusion matrix of naïve Bayes example

		Predicted Class	
		Positive	Negative
Actual Class	Positive	195 (TP)	5 (FN)
	Negative	101 (FP)	99 (TN)

- Performance measures: precision, recall, etc.

9.8 Gaining Insights
Five topics: five-star (left) one-star (right)