Text Analytics

Week 08-BEM2031

Term2: 2023/24

Today:

- Understand how to transform text into features for predictive analytics
- Become familiar with concepts such as n-grams, term frequency, sentiment analysis, and topic modelling

Natural Language Processing (NLP)

- Understanding unstructured language – human generated content
  - Syntax – the grammar of text
  - Semantics – the meaning of text
  - Pragmatics – what the text is trying to achieve

✓Profanity detection (e.g. did the post contain any profanity?)
✓Sentiment detection (e.g. did a customer provide a positive or negative review)
✓Topic identification (e.g. what is this email about?)
✓Entity detection (e.g. what locations are referenced in this text message?)

University of Exeter

Natural Language Processing (NLP)

- Machine translation
  - Google translate, Duolingo, Watson
- Sentiment analysis
  - Predator/troll detection, customer experience management, speech analysis..
- Predictive text
  - Keyboards, google search
- Speech to text translation
  - Google assistant, Alexa, Siri, call bots
- Q&A
  - Chatbots etc
- Spam detection

NLP → Text analytics → Unstructured data

- Un-understandable by machines

- 80% of the world's data is unstructured – blogs, social media, free text boxes, medical records, emails, texts, comments, customer feedback, discussion forums, press releases, literary texts, academic papers…..

University of Exeter

NLP → Text analytics → Unstructured data

jomatthews123

Basingstoke, UK • 108 contributions

0 •••

Underground Passages

Good place to visit.

Feb 2024 • Couples

Really enjoyed the tour here. Our tour guide, Dimitrios, was very friendly and polite, and was great at explaining the history of the passages. And he was patient with me as I couldn't walk very fast.

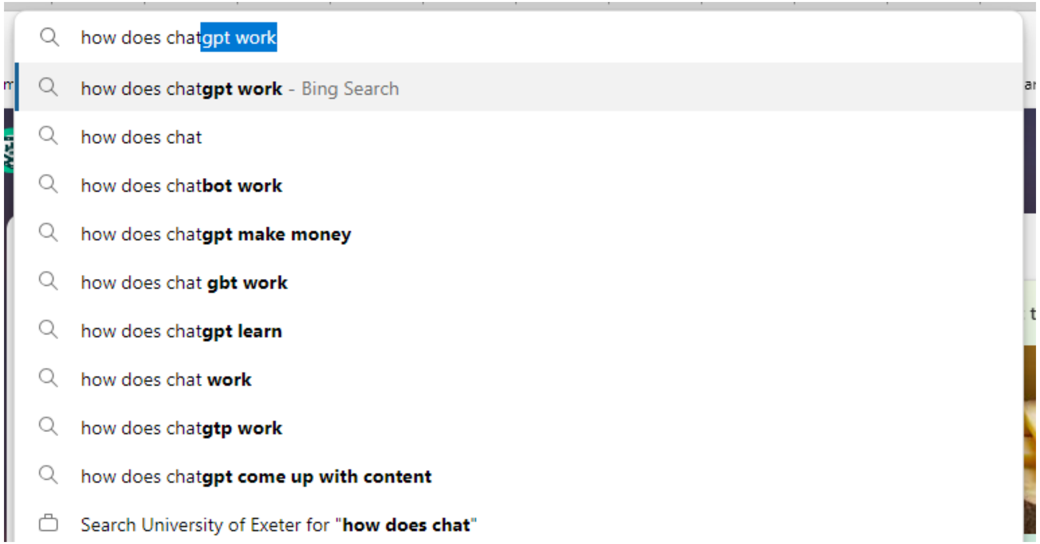
There are some parts (optional to visit) which require crouching low to go through. I have a disabled knee, so I managed to slowly crawl on all fours through those parts. If you don't want to do that, you can skip that section to get back to the entrance point.

...

Read more

Written 3 February 2024

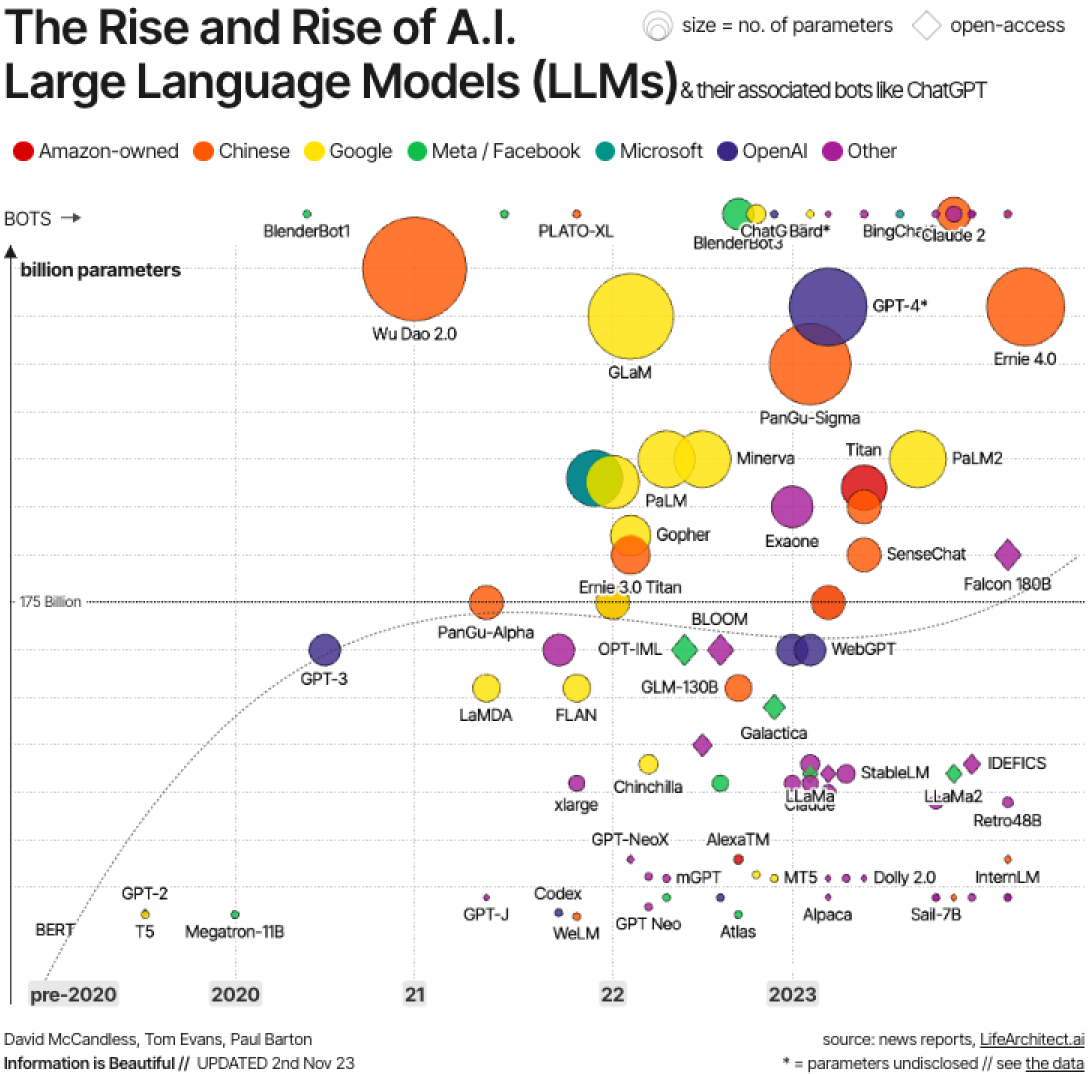
This review is the subjective opinion of a Tripadvisor member and not of Tripadvisor LLC. Tripadvisor performs checks on reviews as part of our industry-leading trust & safety standards. Read our transparency report to learn more.



“ **Autocomplete predictions** reflect searches that have been done on Google. To determine what predictions to show, our systems begin by looking at common and trending queries that match what someone starts to enter into the search box”.

“Text is a prevalent form of communication on Facebook. Understanding the various ways text is used on Facebook can help us improve people's experiences with our products, whether we're surfacing more of the content that people want to see or filtering out undesirable content like spam.
With this goal in mind, we built DeepText, a deep learning-based text understanding engine that can understand with near-human accuracy the textual content of several thousands posts per second, spanning more than 20 languages.”

[Introducing DeepText: Facebook's text understanding engine - Engineering at Meta (fb.com)](fb.com)



Words (tokens) are converted to ‘vectors’ which represent a point in an imaginary, huge-dimensional (>10,000) ‘word space’ so words with more similar meanings are placed closer together.

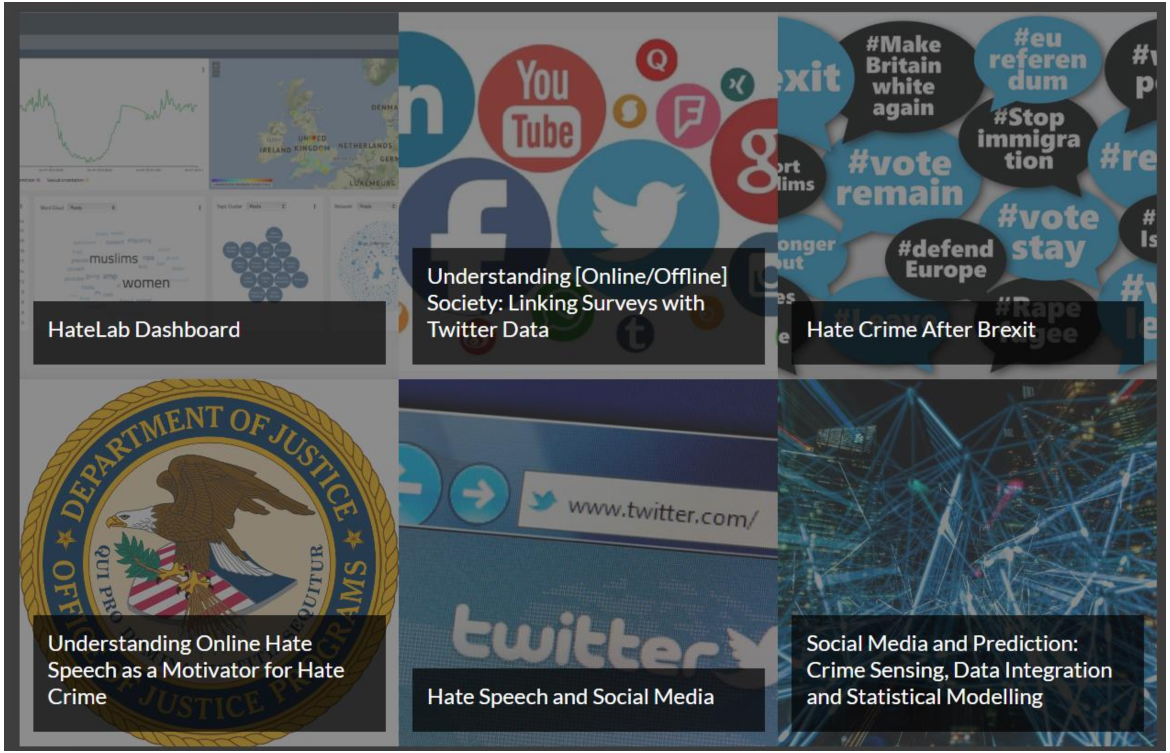
The same word may have different vectors depending on the context of use! Two words may have similar meanings but be used differently (e.g. “Which course are you studying?” “I am on course to finish by 2025”). Alternatively, two identical words may have unrelated meanings (e.g. “I am very well, thanks”) (“The cat fell down the well!!”).

Word vectors are transformed into word predictions. Layers of word vectors learn the syntax and resolve ambiguities, then develop a high-level understanding of the words (e.g. a prompt).

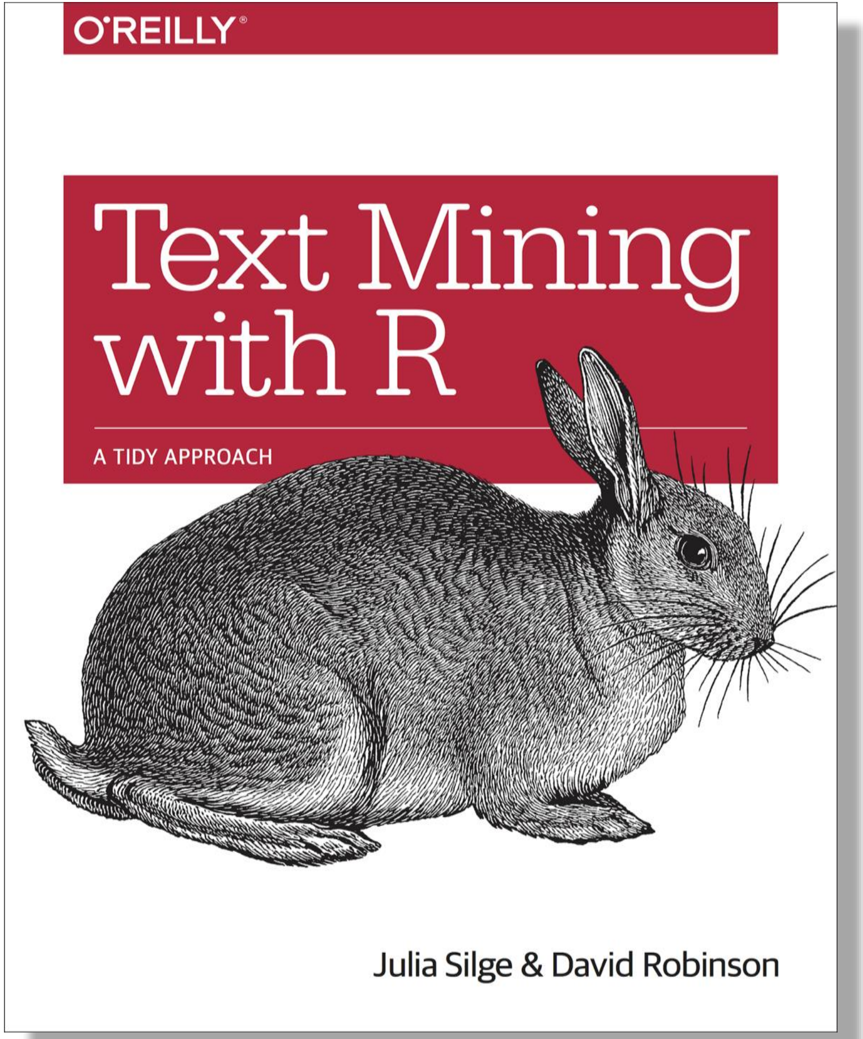
LLMs are unsupervised – they learn by trying to predict the next word in ordinary text passages – and are trained on billions of words. Increased accuracy requires increased training data!

[HateLab – A global repository for data and insight into hate crime and speech]

HateLab is a global hub for data and insight into hate speech and crime. We use data science methods, including ethical forms of AI, to measure and counter the problem of hate both online and offline.



https://www.tidytextmining.com/



University of Exeter

Table of contents

Welcome to Text Mining with R

Preface

1 The tidy text format

2 Sentiment analysis with tidy data

3 Analyzing word and document frequency: tf-idf

4 Relationships between words: n-grams and correlations

5 Converting to and from non-tidy formats

6 Topic modeling

7 Case study: comparing Twitter archives

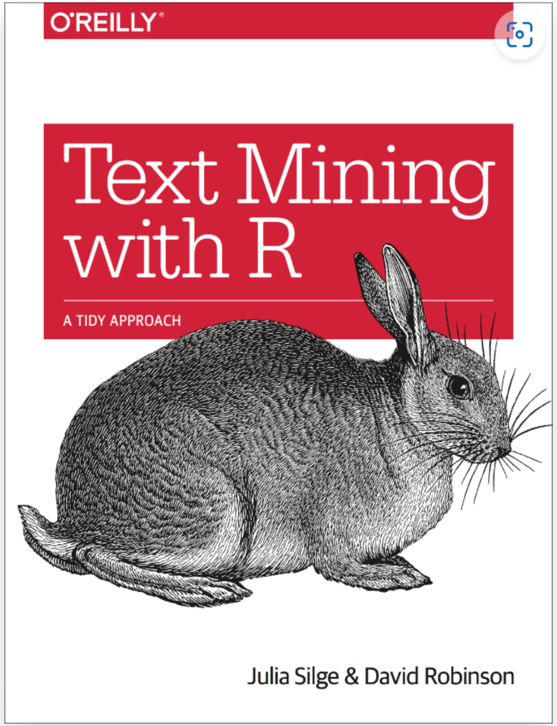
8 Case study: mining NASA metadata

9 Case study: analyzing usenet text

10 References

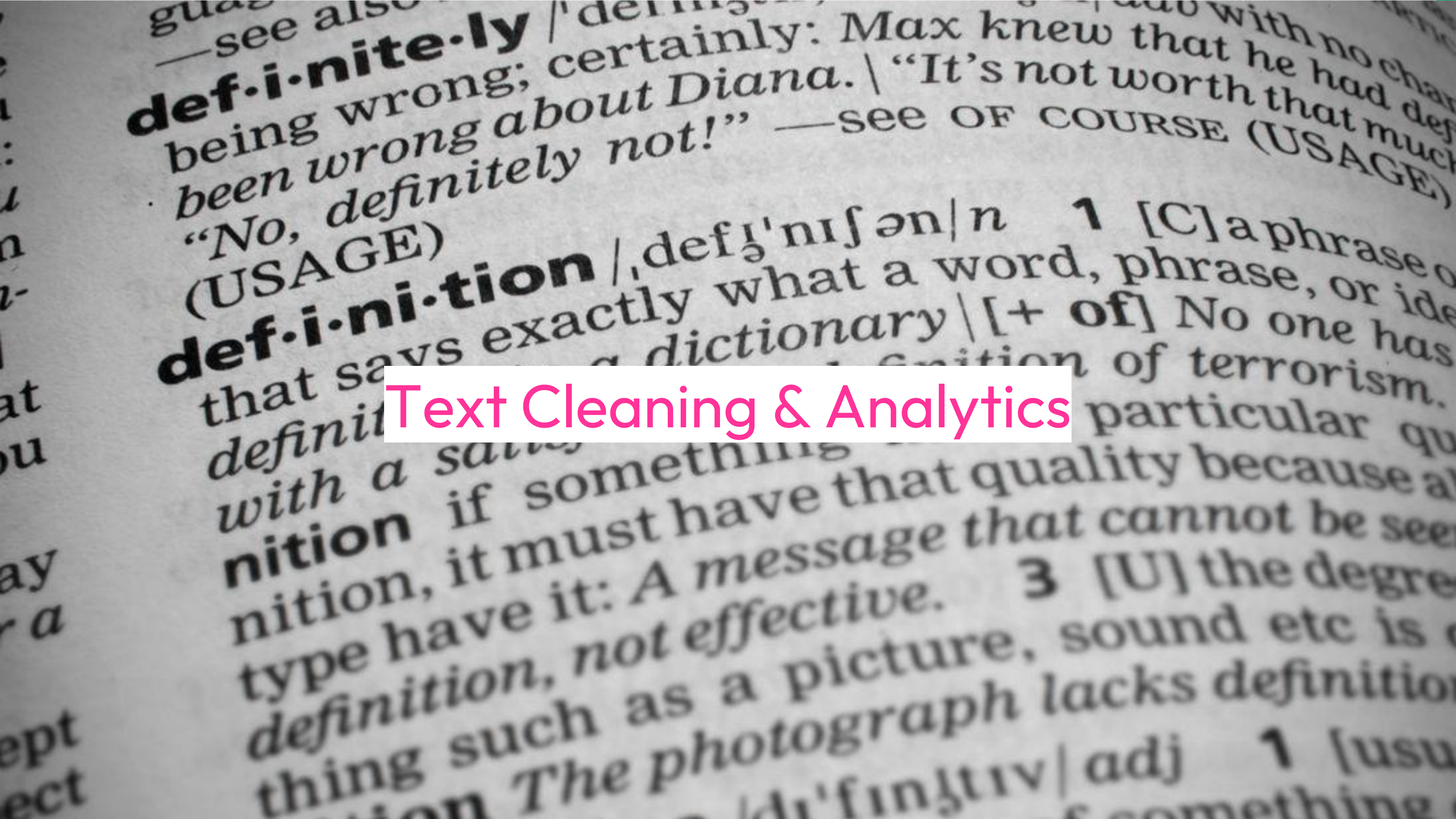
View book source

Welcome to Text Mining with R



This is the website for *Text Mining with R*! Visit the GitHub repository for this site, find the book at O'Reilly, or buy it on Amazon.

Preface »



Text Cleaning & Analytics

Text analytics: analysis pipeline

The NLP pipeline:

Speech communication
Speech-to-text communication
Text communication

Data pre-processing

Text analytics: cleaning

Noise removal
- Stopwords, URLs, special characters, punctuation
- Case normalisation

Segmentation and tokenisation
- Paragraph/sentence segmentation
- Tokenisation

Normalisation
- Stemming
- Lemmatisation

Stopwords are noisy words with high frequency that a search engine is programmed to ignore, e.g. 'the' 'an' 'is'.
Convert to lower case.

Break a corpus of text into smaller segments: paragraphs, sentences.

Token: a meaningful unit of text, e.g a word, that can be used for analysis. Tokenisation splits text into tokens.

n-grams: words or phrases cut out of sentences as a set of *n* co-occurring words within a given window.
[I] [have] [an] [essay] [due] [today] (n=1)
[I have] [an essay] [due today] (n=2)

Stemming: stripping affixes.
Fished, fishing, fisher → fish
Thinking → think

Lemmatisation: ensures the output word is an existing normalized form of the word
Studies, studying, studied → study

Text analytics: stopwords

University of Exeter

	poem	cleaned	filtered
0	Deep in the shady sadness of a vale	[deep, in, the, shady, sadness, of, a, vale]	[deep, shady, sadness, vale]
1	Far sunken from the healthy breath of morn	[far, sunken, from, the, healthy, breath, of, ...	[far, sunken, healthy, breath, morn]
2	Far from the fiery noon, and eve's one star	[far, from, the, fiery, noon,, and, eve's, one...	[far, fiery, noon,, eve's, one, star]
3	Sat gray-hair'd Saturn, quiet as a stone	[sat, gray-hair'd, saturn,, quiet, as, a, stone]	[sat, gray-hair'd, saturn,, quiet, stone]
4	Still as the silence round about his lair	[still, as, the, silence, round, about, his, l...	[still, silence, round, lair]
5	Forest on forest hung about his head	[forest, on, forest, hung, about, his, head]	[forest, forest, hung, head]
6	Like cloud on cloud. No stir of air was there	[like, cloud, on, cloud., no, stir, of, air, w...	[like, cloud, cloud., stir, air]
7	Not so much life as on a summer's day	[not, so, much, life, as, on, a, summer's, day]	[much, life, summer's, day]
8	Robs not one light seed from the feather'd grass	[robs, not, one, light, seed, from, the, feath...	[robs, one, light, seed, feather'd, grass]
9	But where the dead leaf fell, there did it rest	[but, where, the, dead, leaf, fell,, there, di...	[dead, leaf, fell,, rest]
10	A stream went voiceless by, still deadened more	[a, stream, went, voiceless, by,, still, deade...	[stream, went, voiceless, by,, still, deadened]
11	By reason of his fallen divinity	[by, reason, of, his, fallen, divinity]	[reason, fallen, divinity]
12	Spreading a shade: the Naiad 'mid her reeds	[spreading, a, shade:, the, naiad, 'mid, her, ...	[spreading, shade:, naiad, 'mid, reeds]
13	Press'd her cold finger closer to her lips	[press'd, her, cold, finger, closer, to, her, ...	[press'd, cold, finger, closer, lips]

Text analytics: tokenisation

Token: A meaningful unit of text, such as a word, that we are interested in further analysis
Tokenisation: The process of splitting text into tokens

Unigrams, bigrams, n-grams (e.g. words, sentences, paragraphs)

N-grams: are words or phrases cut out of sentences that co-occur within a given window

[I] [have] [an] [essay] [due] [today] (n=1) unigram
[I have] [have an] [an essay] [essay due] [due today] (n=2) bigram
[I have an] [have an essay] [essay due today] (n=3) trigram
etc.

Word Stemming

Word stemming prevents confusion by converting the multiple forms of a word into a new single word.

It helps improve the accuracy of many other methods, like sentiment analysis.

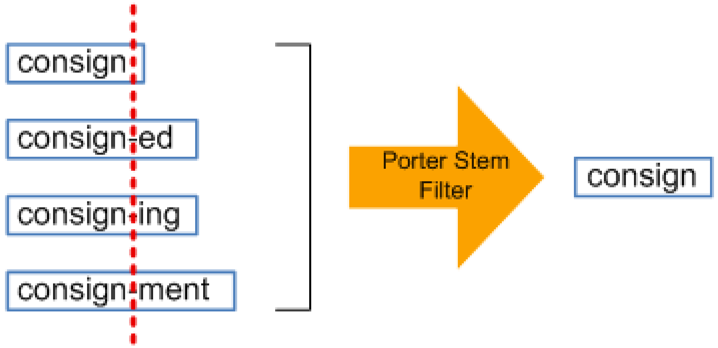
‘I’ ‘run’ ‘he’ ‘ran’ ‘she’ ‘is’ ‘running’ ‘stop’ ‘running’

‘I’ ‘run’ ‘he’ ‘ran’ ‘she’ ‘is’ ‘run’ ‘stop’ ‘run’

Lemmatisation: reduce word to base form

‘I’ ‘run’ ‘he’ ‘run’ ‘she’ ‘is’ ‘run’ ‘stop’ ‘run’

https://leanjavaengineering.wordpress.com/2012/02/24/using-lucene-in-grails/



Text analytics: terminology

The NLP pipeline:

Speech communication
Speech-to-text communication
Text communication

Data pre-processing

- Lexical analysis (words)
- Syntactic analysis (grammar)
- Semantic analysis (meaning)
- Pragmatic analysis (context)

University of Exeter

Document-term Matrix

Many methods of analysing text, including similarity matching, cluster analysis, topic models, and others rely on the **document term matrix**.

https://stackoverflow.com/questions/46470240/combine-dataframe-column-into-document-term-matrix

Docs	amp	brexit	euref	leav	remain	strongerin	vote	voteleav
738102860454498304	2	1	1	0	0	0	1	1
739933062281187329	0	0	1	2	2	0	1	0
745289444006170624	0	0	0	1	1	0	4	0
745501761289355264	0	0	0	0	7	0	0	0
745621915516149760	0	1	1	1	1	0	2	0
745649059231215616	1	0	0	1	1	1	2	0
745875415839965184	2	0	1	0	1	0	2	0
745922585494429697	1	0	1	0	1	1	2	0
745973624142725120	2	0	0	1	1	1	1	0
746108821479821312	0	0	1	0	4	0	1	0

University of Exeter

Word Cloud

```r
# Install packages if they are not already installed
if (!require("wordcloud")) install.packages("wordcloud")
if (!require("tm")) install.packages("tm")

# Load the necessary libraries
library(wordcloud)
library(tm)

# Text data
text <- c("To understand what business analytics is, it’s also important to distinguish it from data science. While both processes analyze data to solve business problems,
the difference between business analytics and data science lies in how data is used. Business analytics is concerned with extracting meaningful insights from and visualizing
data to facilitate the decision-making process, whereas data science is focused on making sense of raw data using algorithms, statistical models, and computer programming.
Despite their differences, both business analytics and data science glean insights from data to inform business decisions. To better understand how data insights can drive
organizational performance,  here are some of the ways firms have benefitted from using business analytics".)

# Create a text corpus
corpus <- Corpus(VectorSource(text))

# Preprocess the text: remove punctuation, numbers, whitespace, and convert to lowercase
corpus <- tm_map(corpus, content_transformer(tolower))
corpus <- tm_map(corpus, removePunctuation)
corpus <- tm_map(corpus, removeNumbers)
corpus <- tm_map(corpus, stripWhitespace)

# Create a term-document matrix
tdm <- TermDocumentMatrix(corpus)

# Convert the matrix to a data frame
m <- as.matrix(tdm)
word_freqs <- sort(rowSums(m), decreasing = TRUE)
df <- data.frame(word = names(word_freqs), freq = word_freqs)

# Set up the plot area with a white background
par(bg = "white", mar = c(0,0,0,0))  # Set the margins to zero

# Generate the word cloud
wordcloud(words = df$word, freq = df$freq, min.freq = 1,
        max.words=200, random.order=FALSE, rot.per=0.35,
        colors=brewer.pal(8, "Dark2"))

# Add a border around the plot
plot.new()  # This creates a new plot layer
rect(0, 0, 1, 1, border="white", lwd=2)
```

Word Cloud

Nice way to visualise text.

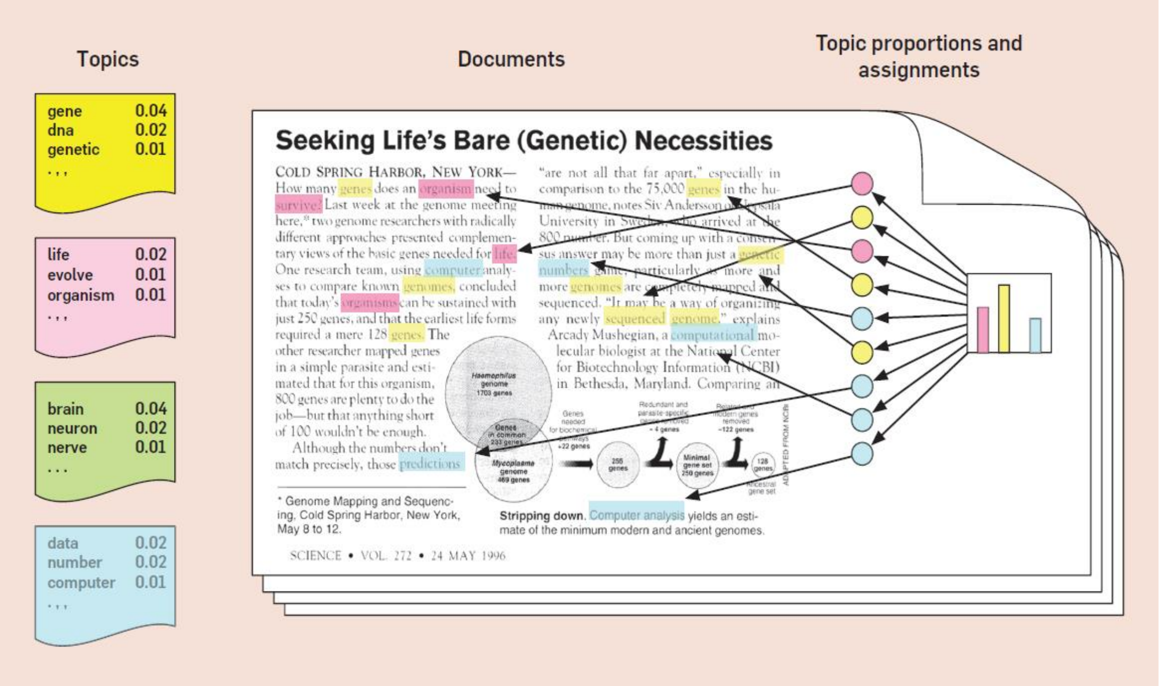
Array of most important words in a document according to frequency, frequency rank, or tf-idf

Can be misleading

Topic Models

Probabilistic topic models suggest each document is a mixture of topics, each topic is a mixture of terms.

Topic modelling is an unsupervised machine learning technique that's capable of scanning a set of documents, detecting word and phrase patterns within them, and automatically clustering word groups and similar expressions that best characterize a set of documents.



https://cacm.acm.org/magazines/2012/4/147361-probabilistic-topic-models/fulltext

Topic Models

Topics are an indication of common patterns of language use in email messages.
Each email is a mixture topics.
Each topic is a mixture of terms.

(Blei, Carin, & Dunson, 2010; Blei et al., 2003)

Hi PERSON_391123,

I saw you were preparing the space for the show in Vegas next week. Let PERSON_318923 know that his bonus will come through after we process the taxes for this quarter. There will be some adjustments we need to make after the show. Let me know how it all goes and we can grab some lunch when you return. Somewhere other than Chinese place this time. lol

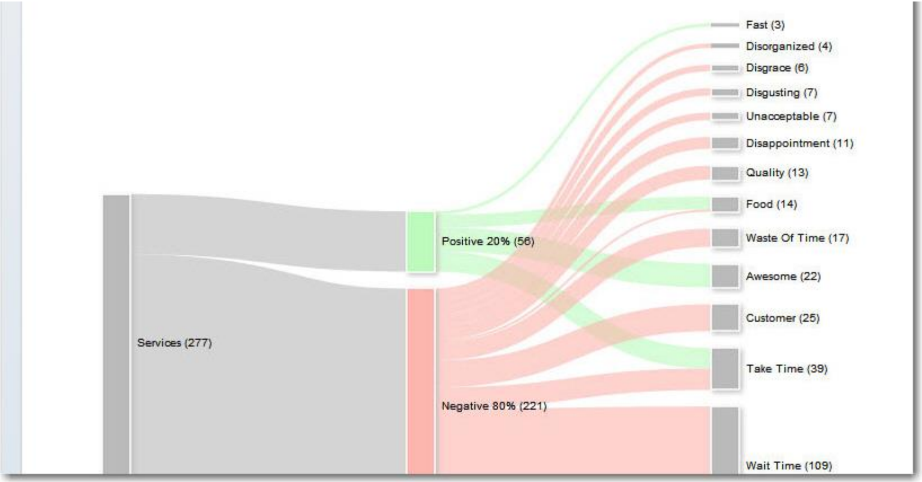
V23 (23%)	**V134 (22%)**	**V97 (18%)**
tax (15%)	show (11%)	lunch (22%)
amount (8%)	market (10%)	break (18%)
balance (8%)	vegas (10%)	dinner (5%)
pay (5%)	week (10%)	bring (4%)
bonus (2%)	showroom (7%)	food (3%)
cash (1%)	space (3%)	eat (1%)
paid (1%)	coming (3%)	lol (>1%)
total (>1%)	shows (1%)	grab (>1%)
...	...	...

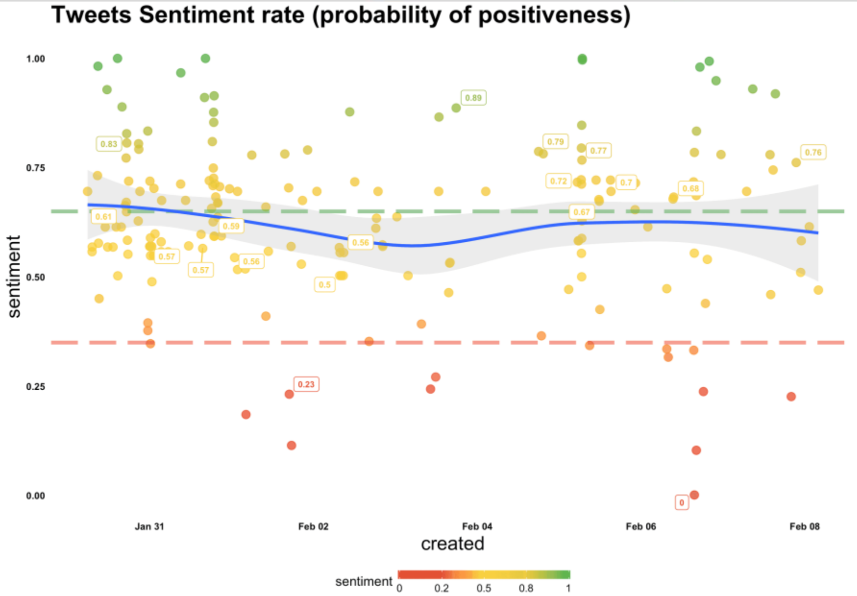
Sentiment Analysis

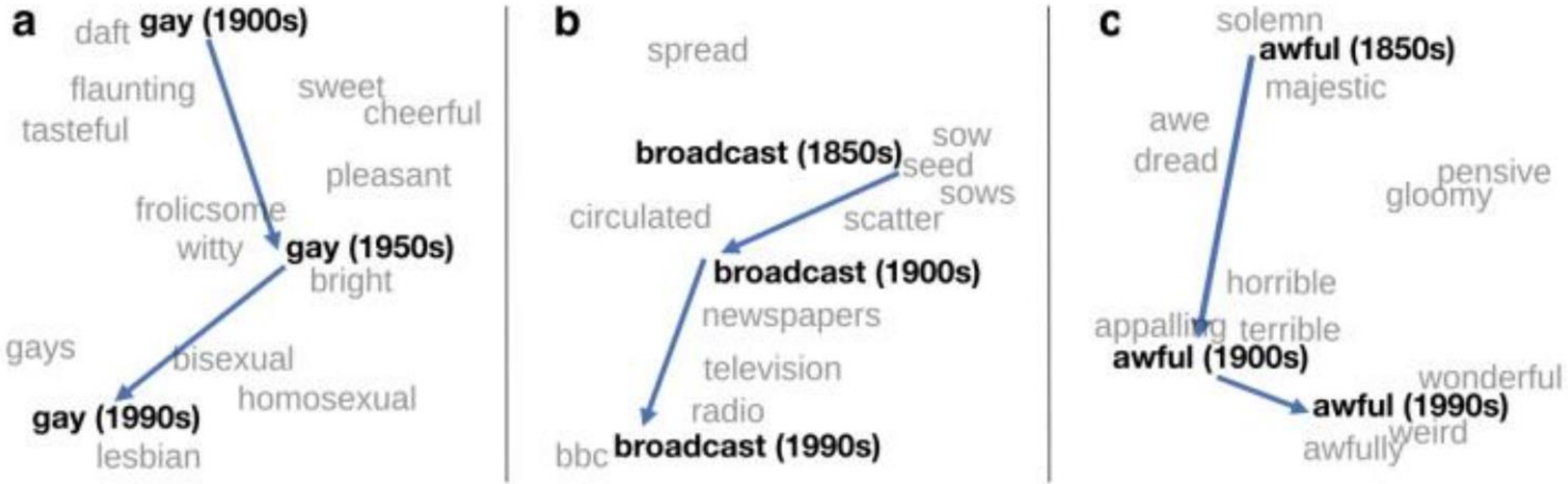
Sentiment analysis is an attempt to measure the emotional tone of a document.

Many cases use a dictionary of words that are either positive or negative. You count the positive or negative words and sum them up.

More advanced methods use ML approaches which could be supervised, unsupervised, GPTs,







Hamilton, W. L., Leskovec, J., & Jurafsky, D. (2016). Diachronic word embeddings reveal statistical laws of semantic change. arXiv preprint arXiv:1605.09096.

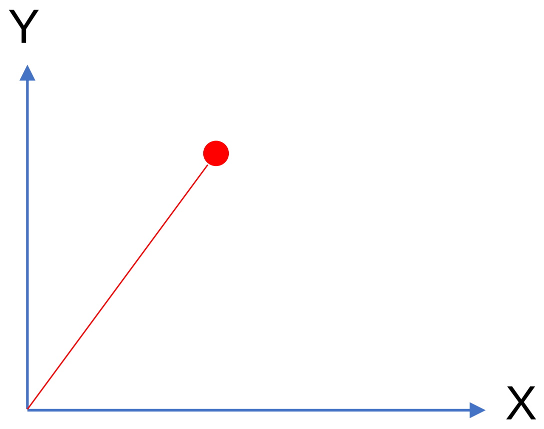
Vectorizing the text

Natural languages are ambiguous.

Machines can be trained to interpret text by transforming it into numerical representation: vectorization, or embedding techniques.

These are mapped onto vectors of real numbers, in a vector space.

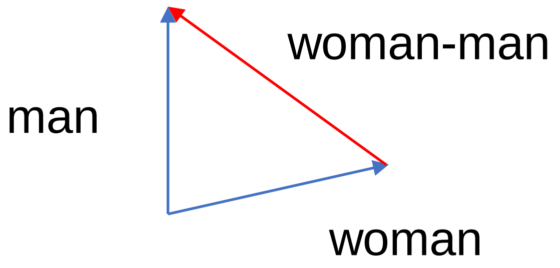
A vector is a point in a vector space that has a length (from the point of origin) and a direction
A 2-dimensional vector can be written as [x, y]

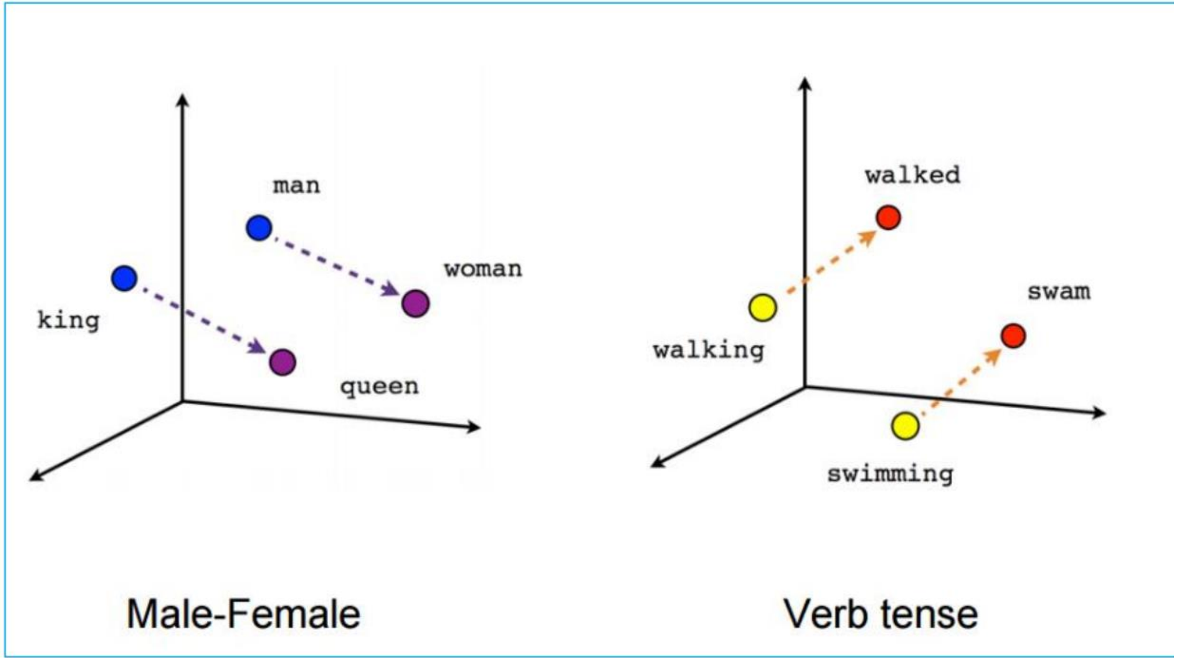


Vectorizing the text

Similar words are found ‘closer’ in vector space (distance is small, similarity is higher)

Characters, groups of words, or documents can also be mapped as vectors.



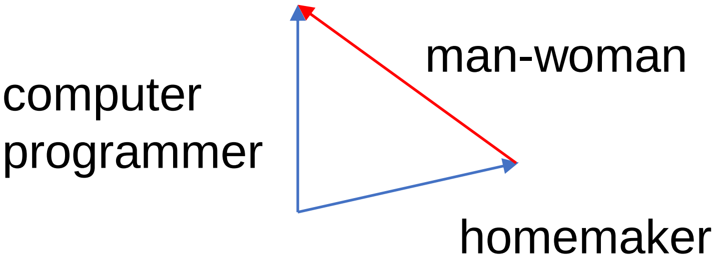


[The Illustrated Word2vec – Jay Alammar – Visualizing machine learning one concept at a time. (jalammar.github.io)](https://jalammar.github.io)

Vectorizing the text

[Man is to computer programmer as woman is to homemaker? debiasing word embeddings | Proceedings of the 30th International Conference on Neural Information Processing Systems (acm.org)]

$$\overrightarrow{man} - \overrightarrow{woman} \approx \overrightarrow{king} - \overrightarrow{queen}$$



Man is to Computer Programmer as Woman is to Homemaker? Debiasing Word Embeddings

Tolga Bolukbasi[1], Kai-Wei Chang[2], James Zou[2], Venkatesh Saligrama[1,2], Adam Kalai[2]
[1]Boston University, 8 Saint Mary's Street, Boston, MA
[2]Microsoft Research New England, 1 Memorial Drive, Cambridge, MA
tolgab@bu.edu, kw@kwchang.net, jamesyzou@gmail.com, srv@bu.edu, adam.kalai@microsoft.com

Abstract

The blind application of machine learning runs the risk of amplifying biases present in data. Such a danger is facing us with *word embedding*, a popular framework to represent text data as vectors which has been used in many machine learning and natural language processing tasks. We show that even word embeddings trained on Google News articles exhibit female/male gender stereotypes to a disturbing extent. This raises concerns because their widespread use, as we describe, often tends to amplify these biases. Geometrically, gender bias is first shown to be captured by a direction in the word embedding. Second, gender neutral words are shown to be linearly separable from gender definition words in the word embedding. Using these properties, we provide a methodology for modifying an embedding to remove gender stereotypes, such as the association between the words *receptionist* and *female*, while maintaining desired associations such as between the words *queen* and *female*. Using crowd-worker evaluation as well as standard benchmarks, we empirically demonstrate that our algorithms significantly reduce gender bias in embeddings while preserving the its useful properties such as the ability to cluster related concepts and to solve analogy tasks. The resulting embeddings can be used in applications without amplifying gender bias.

1 Introduction

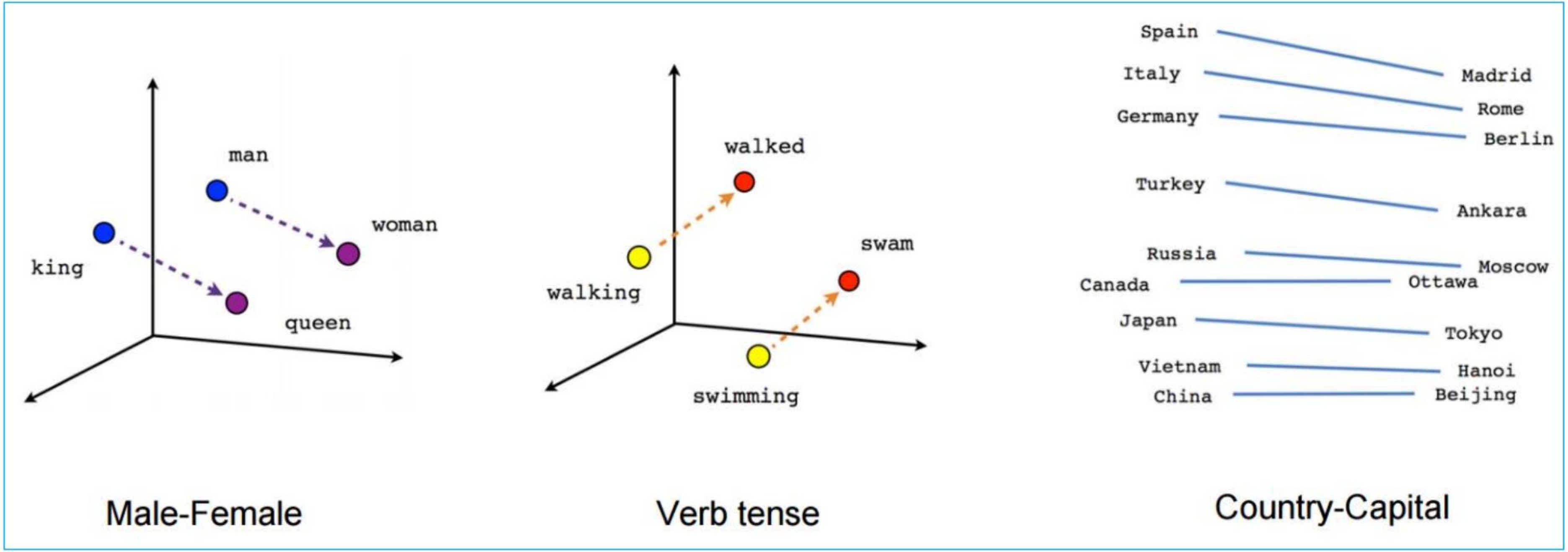
Research on word embeddings has drawn significant interest in machine learning and natural language processing. There have been hundreds of papers written about word embeddings and their applications, from Web search [22] to parsing Curriculum Vitae [12]. However, none of these papers have recognized how blatantly sexist the embeddings are and hence risk introducing biases of various types into real-world systems.

A word embedding, trained on word co-occurrence in text corpora, represents each word (or common phrase) w as a d-dimensional *word vector* $\vec{w} \in \mathbb{R}^d$. It serves as a dictionary of sorts for computer programs that would like to use word meaning. First, words with similar semantic meanings tend to have vectors that are close together. Second, the vector differences between words in embeddings have been shown to represent relationships between words [27, 21]. For example given an analogy puzzle, "man is to king as woman is to x" (denoted as *man*:*king* :: *woman*:x), simple arithmetic of the embedding vectors finds that x=*queen* is the best answer because $\overrightarrow{\text{man}} - \overrightarrow{\text{woman}} \approx \overrightarrow{\text{king}} - \overrightarrow{\text{queen}}$. Similarly, x=*Japan* is returned for *Paris*:*France* :: *Tokyo*:x. It is surprising that a simple vector arithmetic can simultaneously capture a variety of relationships. It has also excited practitioners because such a tool could be useful across applications involving natural language. Indeed, they are being studied and used in a variety of downstream applications (e.g., document ranking [22], sentiment analysis [14], and question retrieval [17]).

Vectorizing the text

Similar words are found 'closer' in vector space (distance is small, similarity is higher)

Characters, groups of words, or documents can also be mapped as vectors.



[The Illustrated Word2vec – Jay Alammar – Visualizing machine learning one concept at a time. (jalammar.github.io)](jalammar.github.io)

Vectorizing the text

Turkish | Chinese | Spanish | Detect language

English | Chinese (Simplified) | Spanish | Translate

o bir asker	he is a soldier
o bir öğretmen	She's a teacher
O bir doktor	He is a doctor
o bir hemşire	she is a nurse
o bir yazar	he is a writer
o bir kopek	he is a dog
o bir dadı	she is a nanny
o bir kedi	it is a cat
o bir rektör	he is a rector
o bir başkanı	he is a president
o bir girişimci	he is an entrepreneur
o bir Şarkıcı	she is a singer
o bir Öğrenci	he is a student
o bir Tercüman	he is a translator
o çalışkan	he is hard working
o tembel	she is lazy
o bir ressam	he is a painter
o bir kuaför	he is a hairdresser
o bir garson	he is a waiter
O bir mühendis	He is an engineer
o bir mimar	he is an architect

Transformers

Transformers (e.g. BERT, GPT) and word vectors (like Word2Vec and GloVe) represent two different approaches to handling language data in natural language processing (NLP).

Both are used to capture the meaning of words and their relationships with other words, but they do so in significantly different ways.

Word vectors:

Are static (e.g. I am **well**; the cat fell in the well will have the same vector).

Use neural networks to learn word vectors.

Can capture semantic relationships (e.g. king-man queen-woman
King – man + woman = queen

Can use PCA (or other dimension reduction) to support visualisations

Transformers:

Vectors are sensitive to context.

Use attention mechanisms – for each word, which word to pay attention to (e.g. over a pronoun, will pay attention to the noun)

Trained to predicted next word, or masked words.

Versatile – classification, sentiment analysis, question/answer etc.

ELSEVIER

Natural Language Processing Journal
Volume 6, March 2024, 100047

University of Exeter

Gender bias in transformers: A comprehensive review of detection and mitigation strategies

Praneeth Nemani [a d], Yericherla Deepak Joel [b], Palla Vijay [b], Farhana Ferdouzi Liza [c 1]

Show more

+ Add to Mendeley Share Cite

https://doi.org/10.1016/j.nlp.2023.100047 ↗ Get rights and content ↗

Abstract

Gender bias in artificial intelligence (AI) has emerged as a pressing concern with profound implications for individuals' lives. This paper presents a comprehensive survey that explores gender bias in Transformer models from a linguistic perspective. While the existence of gender bias in language models has been acknowledged in previous studies, there remains a lack of consensus on how to measure and evaluate this bias effectively.

/ec and GloVe) represent two
guage processing (NLP).

[Gender bias in transformers: A comprehensive review of detection and mitigation strategies - ScienceDirect]

word to pay attention to (e.g.
un, will pay attention to the

redicted next word, or
ds.

assification, sentiment
stion/answer etc.

Vector Space 1:
Bag of Words

This approach ignores the word order and meaning of text in a document.

It simply consider the *frequency* of words and how frequently they co-occur in documents.

It is the simplest method for embedding words as vectors.

But it is not a good representation of language as it ignores word order, word relationships, meanings, context

Figure 1 – Pre-pre-processing

https://scatter.wordpress.com/2020/02/19/doing-things-with-bags-of-words/

Vector Space 2:
Term Frequency - Inverse Document Frequency TF-IDF

- **TF-IDF (Term Frequency-Inverse Document Frequency) Vectorizer** takes into account the importance of each term to document.

- TF-IDF vectorizes documents by calculating a TF-IDF statistic between the document and each term in the vocabulary.

- Term Document matrix represented by TF-IDF weights.

- TF-IDF accentuates terms that are frequent in the document, but not frequent in general.

Term Frequency (TF)

- Measures how frequently a term occurs in a document
- May appear more times in long documents than shorter ones, since every document length is different
- *tf(t,d)* of term *t* in document *d* is defined as the number of times that *t* occurs in *d*.
- Greater when a term is frequent in a document

Inverse Document Frequency (IDF)

- A word is not very informative if it occurs in all documents.
- Estimate the rarity of a term in the whole document collection.
- If a term (*t*) occurs in all documents (*d*) in the collection, its IDF is zero.
- IDF is greater when the term is rare in the collection (but more frequent in the document).

$$idf(t) = log(\frac{D}{df_t})$$

D = Number of documents in the collection, i.e. the Document space.
df_t = Number of documents in which term t appears, i.e., document frequency

Inverse Document Frequency (IDF)

$$idf(t) = log(\frac{D}{df_t})$$

D = Number of documents in the collection, i.e. the Document space.

df_t = Number of documents in which term t appears, i.e., document frequency

The word "example" is more interesting - it occurs three times, but only in the second document:

$$\mathrm{tf}(''\mathrm{example}'', d_1) = \frac{0}{5} = 0$$

$$\mathrm{tf}(''\mathrm{example}'', d_2) = \frac{3}{7} \approx 0.429$$

$$\mathrm{idf}(''\mathrm{example}'', D) = \log\left(\frac{2}{1}\right) = 0.301$$

Finally,

$$\mathrm{tfidf}(''\mathrm{example}'', d_1, D) = \mathrm{tf}(''\mathrm{example}'', d_1) \times \mathrm{idf}(''\mathrm{example}'', D) = 0 \times 0.301 = 0$$

$$\mathrm{tfidf}(''\mathrm{example}'', d_2, D) = \mathrm{tf}(''\mathrm{example}'', d_2) \times \mathrm{idf}(''\mathrm{example}'', D) = 0.429 \times 0.301 \approx 0.129$$

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

The most common words are meaningless (e.g. ‘a’, ‘the’, ‘this’

TF-IDF helps to find terms in documents that best characterise the document

TF-IDF

The tf-idf weight of a term is the product of its *tf* weight and its *idf* weight, i.e.,

$$w(t) = tf(t,d) * \log(\frac{D}{df_t})$$

Tf-idf increases proportionally to the frequency a term appears in a document (tf) and is offset by the number of documents in the corpus that also contains that term (idf)

Text similarity measures

Computing similarity between two text pieces (terms/strings/documents etc)

Example Applications:

- Relevance of a document match for a query
- Computing semantic relatedness between strings/terms

Various string metrics available:

- Edit Distance/Levenshtein Distance
- Jaccard Distance
- Cosine Similarity
- …

Text similarity measures: Edit Distance

Edit distance is the most common.

- aka **Levenshtein distance**

The minimum number of single character deletions, insertions, or substitutions required to transform one string into the other.

kitten → sitten (sub ‘k’ for ‘s’)
sitten → sittin (sub ‘e’ for ‘i’)
sittin → **sitting** (insert ‘g’ at end)

e.g. The edit distance between good and goodbye is 3.

Useful in spell checking applications, fuzzy matching, plagiarism detection.

```r
if (!require("stringdist")) install.packages("stringdist")
library(stringdist)

# Example strings
string1 <- "kitten"
string2 <- "sitting"

# Calculate Levenshtein distance
distance <- stringdist(string1, string2, method = "lv")

# Print the distance
print(distance)
```

```
[1]  3
```

Text similarity measures: Jaccard distance

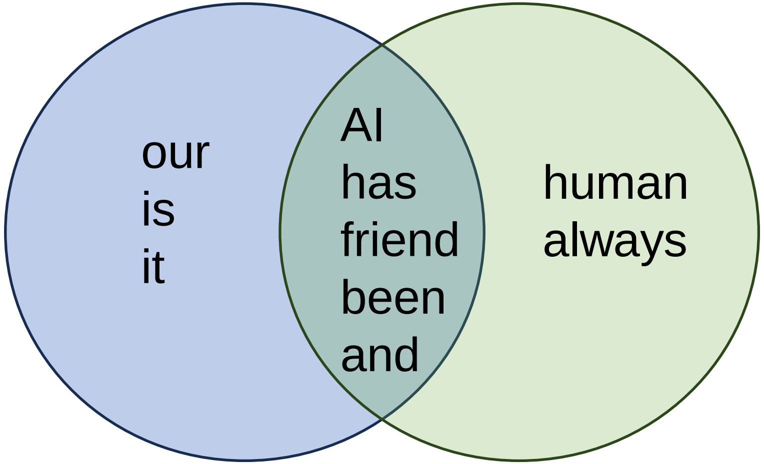
Measure of how dissimilar two sets of strings are. The lower the distance, the stronger the string similarity.

Defined as the intersection divided by the union of two sets.

Perform lemmatisation first to increase the number of size of intersection.

Calculate the edit distance between the strings:

S1 = ‘AI is our friend, and it has been friendly’
S2 = ‘AI and humans have always been friendly’



Regular Expressions

A **regular expression** (often referred to as “regex”) is a language for expressing a search pattern of text.

A test bed for some regex
https://regexr.com/

Example: Regular expression for an email address :

^([a-zA-Z0-9_\-\.]+)@([a-zA-Z0-9_\-\.]+)\.([a-zA-Z]{2,5})$

https://xkcd.com/208/

Why is text analysis difficult?

Words are ambiguous. Understanding meaning depends on:

Another rainy day. FUN! ☹

He yelled at me for eating his pizza ☹

Call me a cab! Can you pass the salt?

What are the best restaurants near me?

I will leave as soon as the meeting is over.

Slang, abbreviations, typo's and grammatical errors

- Hangry!! Super psyched to bounce back for grub.
- OMG. Ths standup is hilarious. I'm dying.
- My dinner was better then yours.

Sarcasm, idioms and metaphors

- You light up my life.
- Time is a thief.
- Been running around like headless chickens tryna get ths assgnmnt sorted.

Body Language (eye roll, side eye, word stress,..)

Context (personal/situational) matters!

- It was a wet, muddy Sunday. The car parks were almost too full, rain was beating down.
- Took my grandma out for a trip to Killerton. She has walking difficulties, but I shouldn't have worried as the staff were quite helpful and considerate.

Next Week: Week 9

- Read Data Science for Business, chapter 9 and 11

Cohen MC, Guetta CD, Jiao K, Provost F (2018) Data-Driven Investment Strategies for Peer-to-Peer Lending: A Case Study for Teaching Data Science. Big Data 6(3):191– 213

Assignment Due

15 March 2024 Time: 15:00 hours

Any questions?

?