Advanced Machine Learning in Finance (COMP0162): Natural Language Processing

Paolo Barucca Tomaso Aste

Department of Computer Science
University College London

6 February 2023

What is Natural Language Processing (NLP)?

- Natural language processing is a type of AI that assists programs in understanding and interpreting human language.
- NLP fills the gap between human communication and computer understanding.
- Examples
  - Amazon's Alexa
  - Google Assistant
  - Customer service chatbots
  - Real-time voice-to-text translators
  - ChatGPT (Generative pre-trained transformers)

Why we need Natural Language Processing?

- ... while we humans are great users of language, we are also very poor at formally understanding and describing the rules that govern language. [2]
- ...understanding natural language require large amounts of knowledge about morphology, syntax, semantics and pragmatics as well as general knowledge about the world. [1]

Benefits of NLP in Finance

- **Automation.** The use of natural language processing (NLP) can replace the manual methods used by financial institutions to convert unstructured data into a more useable format. Automating the recording of earnings calls, management presentations, and acquisition announcements, for example.
- **Data enrichment.** Adding context to unstructured data improves its searchability and actionability. Machine learning can add information to that raw text, indicating areas that deal with environmental, financial, or other topics of importance.

Use cases of NLP in Finance

- Fraud Detection
- Data Entry
- Document Classification
- Customer Personalization
- Risk Assessment
- Customer Service

Basic NLP Tasks

Many natural language processing tasks involve syntactic and semantic analysis, used to break down human language into machine-readable chunks.

- **Syntactic analysis**, also known as parsing or syntax analysis, identifies the syntactic structure of a text and the dependency relationships between words, represented on a diagram called a parse tree.
- **Semantic analysis** focuses on identifying the meaning of language. However, since language is polysemic and ambiguous, semantics is considered one of the most challenging areas in NLP.

NLP Sub-tasks

- **Tokenization** *Customer service couldn't be better! = "customer service" "could" "not" "be" "better".*
- **Part-of-speech tagging (PoS Tagging).** *"Customer service": NOUN, "could": VERB, "not": ADVERB, "be": VERB, "better": ADJECTIVE, "!": PUNCTUATION*
- **Stopword Removal.** *"Hello, I'm having trouble logging in with my new password", becomes "trouble", "logging in", "new", "password".*

NLP Sub-tasks Contd.

- **Lemmatization**. The terms “is, are, am, were, and been,” are grouped under the lemma ‘be’.
- **Stemming**. Stemming the words “consult”, “consultant”, “consulting”, and “consultants” would result in the root form “consult.”
  Stemmers can lead to less-accurate results, they are easier to build and perform faster than lemmatizers.

NLP Sub-tasks Contd.

- **Word Sense Disambiguation.** *You should read this book; it's a great novel!* and *You should book the flights as soon as possible.*
- **Named Entity Recognition (NER).** In the phrase *"Susan lives in Los Angeles"*, a *person (Susan)* is related to a *place (Los Angeles)* by the semantic category *"lives in"*.

Challenge

"To fully comprehend human language, NLP tools need to learn to look beyond definitions and word order, to understand context, word ambiguities, and other complex concepts connected to messages. "

Sentiment Analysis

"I am happy with this water bottle."

Positive

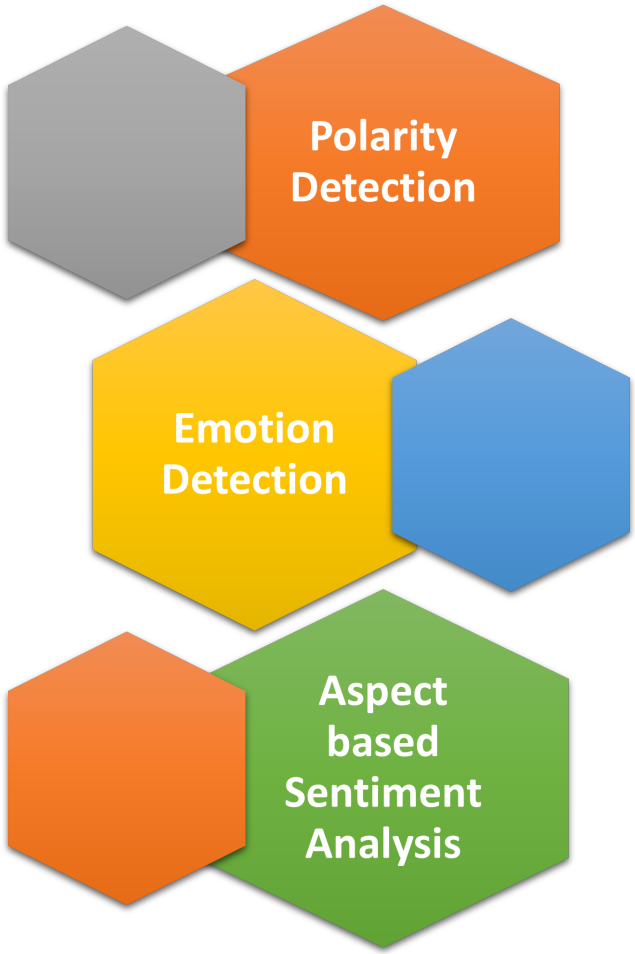
"This is a bad investment."

Negative

"I am going to walk today."

Neutral

Types of Sentiment Analysis



Methods: Lexicon Based Approach

A set of rules based on sentiment lexicons are used to classify the words into either positive or negative along with their corresponding intensity measure.

Example:

Consider this problem instance: “Sam is a great guy.”

Tokens: ’Sam’, ’is’, ’a’, ’great’, ’guy’, ’.’

Removing stop words and punctuation: ’Sam’, ’great’, ’guy’

Identify polarity using some lexicons: ’Sam’: ’Neutral’, ’great’: ’Positive’, ’guy’: ’Neutral’

Calculating the score:

$$score = \frac{positive\_count - negative\_count}{word\_count}$$
$$score = \frac{1 - 0}{3} = 0.33 \quad (1)$$

Available Resources

Challenges

- The financial domain is characterized by a unique vocabulary, which calls for domain-specific sentiment analysis.
  For example, “liability” is generally a negative word, but in the financial domain it has a neutral meaning. Furthermore, “bull” is neutral in general, but in finance, it is strictly positive, while “bear” is neutral in general, but negative in finance.
  Solution: Using a domain specific Lexicon Directory, for example: Loughran and McDonald [3]
- The main drawback with the rule-based approach for sentiment analysis is that the method only cares about individual words and completely ignores the context in which it is used.
  Solution: Using predictive models, for example, TF-IDF, and Word2Vec.

Method: Word Embeddings

Word embedding is the collective name for a set of language modeling and feature learning techniques in natural language processing (NLP) where words or phrases from the vocabulary are mapped to vectors of real numbers.
Word embeddings are a type of word representation that allows words with similar meaning to have a similar representation, naturally capturing their meaning.

Why Word Embeddings are needed?

Many Machine Learning algorithms and almost all Deep Learning Architectures are incapable of processing strings or plain text in their raw form. They require numbers as inputs to perform any sort of job, be it classification, regression etc. in broad terms.

Types of Word Embeddings

- Frequency based Embeddings
  - Bag of Words
  - TF-IDF Vectors
  - Co-occurrence Vectors
- Prediction based Embeddings
  - Context-insensitive Word Embeddings
    - Word2Vec
    - Global Vectors (GloVe)
    - FastText
  - Context-sensitive Word Embeddings
    - Embeddings from Language Models (ELMO)
    - Bidirectional Encoder Representations from Transformers (BERT)

Bag of Words (BoW)

- A bag-of-words model, or BoW for short, is a way of extracting features from text for use in modeling, such as with machine learning algorithms.
- A bag-of-words is a representation of text that describes the occurrence of words within a document. It involves two things:
  - A vocabulary of known words.
  - A measure of the presence of known words.
- The model is only concerned with whether known words occur in the document, not where in the document.

Example

Documents	Tokens						
	it	was	the	age	of	wisdom	foolishness
D_1 = 'it was the age of wisdom'	1	1	1	1	1	1	0
D_2 = 'it was the age of foolishness'	1	1	1	1	1	0	1

Vectors for the documents:

- $D_1 = [1,1,1,1,1,1,0]$
- $D_2 = [1,1,1,1,1,0,1]$

Here we use binary scoring of the presence or absence of words. Some additional simple scoring methods include:

- **Counts.** Count the number of times each word appears in a document.
- **Frequencies.** Calculate the frequency that each word appears in a document out of all the words in the document.

Challenge

A problem with scoring word frequency is that highly frequent words start to dominate in the document (e.g. larger score), but may not contain as much "informational content" to the model as rarer but perhaps domain specific words.

Discarding word order ignores the context, and in turn meaning of words in the document (semantics). Context and meaning can offer a lot to the model, that if modeled could tell the difference between the same words differently arranged ("this is interesting" vs "is this interesting"), synonyms ("old bike" vs "used bike"), and much more.

Term frequency-Inverse document frequency (TF-IDF)

TF-IDF rescales the frequency of words by how often they appear in all documents, so that the scores for frequent words like “the” that are also frequent across all documents are penalized.

- **Term Frequency:** is a scoring of the frequency of the word in the current document.
- **Inverse Document Frequency:** is a scoring of how rare the word is across documents.

The scores have the effect of highlighting words that are distinct (contain useful information) in a given document.

Term Frequency (tf): gives us the frequency of the word in each document in the corpus. It is the ratio of number of times the word appears in a document compared to the total number of words in that document. Each document has its own tf.

$$tf(t,d) = \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}} \tag{2}$$

Inverse Data Frequency (idf): used to calculate the weight of rare words across all documents in the corpus. The words that occur rarely in the corpus have a high IDF score.

$$idf(t) = log(\frac{N}{||\{d \in D : t \in d\}||}) \tag{3}$$

$$tfidf(t) = tf(t,d) * idf(t)$$

Example

Terms	TF		IDF(t)	TF*IDF	
	D_1	D_2		D_1	D_2
it	1/7	1/7	log(2/2) = 0	0	0
was	1/7	1/7	log(2/2) = 0	0	0
the	2/7	2/7	log(2/2) = 0	0	0
age	1/7	1/7	log(2/2) = 0	0	0
of	1/7	1/7	log(2/2) = 0	0	0
wisdom	1/7	0	log(2/1) = 0.3	0.043	0
foolishness	0	1/7	log(2/1) = 0.3	0	0.043

Vectors for the documents:

- D_1 = [0,0,0,0,0,0.043,0]
- D_2 = [0,0,0,0,0,0,0.043]

Challenges

- TF-IDF is based on the bag-of-words (BoW) model, therefore it does not capture position in text, semantics, co-occurrences in different documents, etc.
- For this reason, TF-IDF is only useful as a lexical level feature
- Cannot capture semantics (e.g. as compared to topic models, word embeddings)

Prediction based Word Embeddings

- Prediction based methods generally train on neural network to predict the next word, given a context.
- Learning word embeddings from scratch is a challenging problem due to two primary reasons:
  - Sparsity of training data
  - Large number of trainable parameters

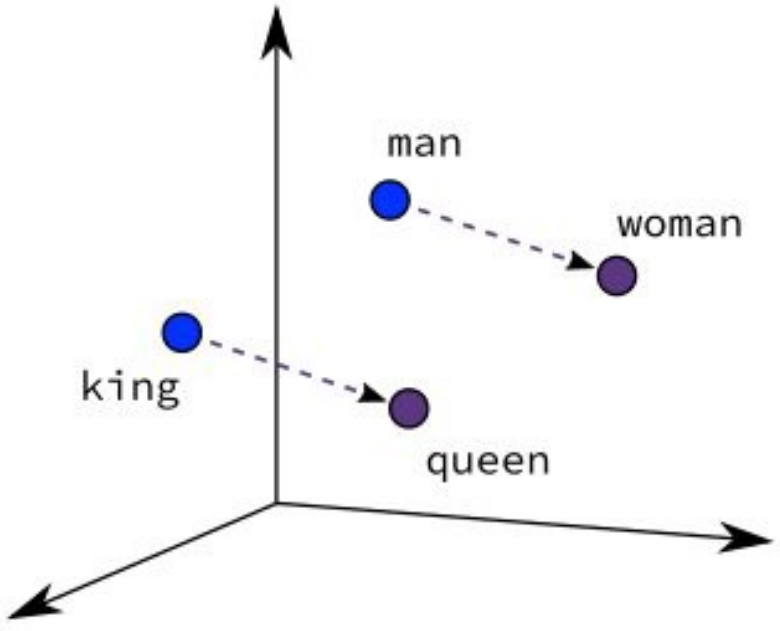
Word2Vec

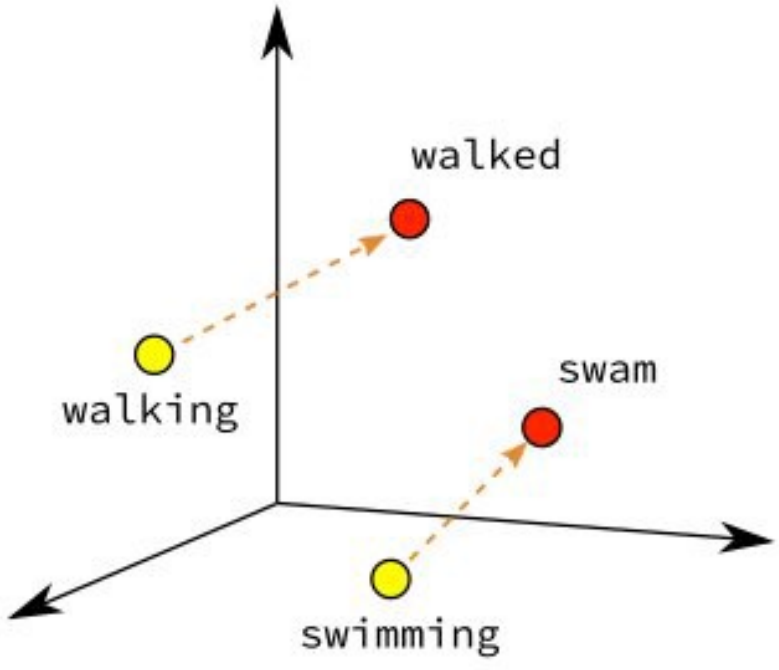
- Word2Vec is trained on the Google News dataset (about 100 billion words). It has several use cases such as Recommendation Engines, Knowledge Discovery, and also applied in the different Text Classification problems.
- The architecture of Word2Vec is a feed-forward neural network with just one hidden layer. Hence, it is sometimes referred to as a Shallow Neural Network architecture.
- *Use running text as implicitly supervised training data!*

Word2Vec

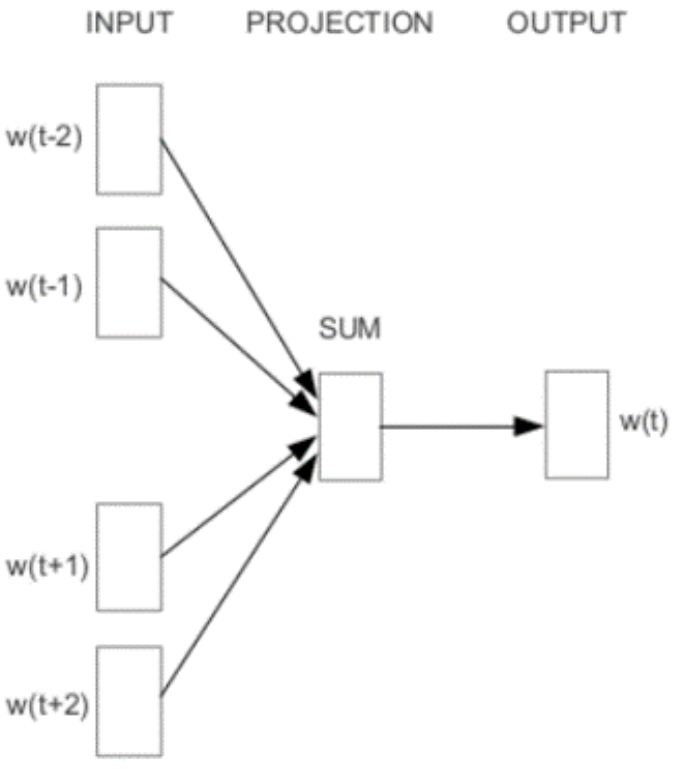
- Depending on the way the embeddings are learned, Word2Vec is classified into two approaches
  - Continuous Bag-of-Words (CBOW)
  - Skip-gram model
- Continuous Bag-of-Words (CBOW) model learns the focus word given the neighboring words whereas the Skip-gram model learns the neighboring words given the focus word. That's why, Continous Bag Of Words and Skip-gram are inverses of each other

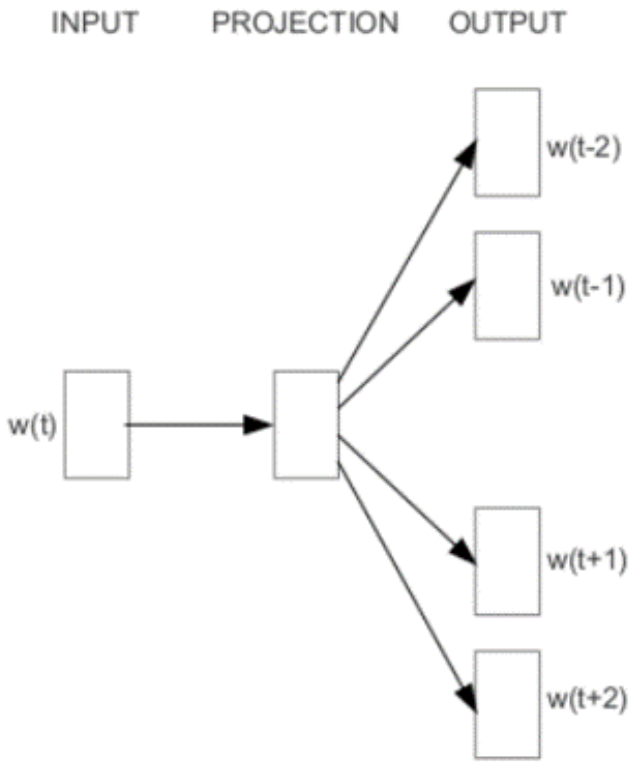
Word Analogies





Word2Vec Archtiecture





Context Window

“I have failed at times but I never stopped trying”.

To learn the embedding of the word “failed”. The **focus word** is **“failed”**.

A **context window** refers to the number of words appearing on the left and right of a focus word. The words appearing in the context window are known as **neighboring words** (or context).

Let’s fix the context window to 2 and then input and output pairs for both approaches.

Continuous Bag-of-Words (CBOW):

Input = [I, have, at, times],

Output = failed

Skip-gram:

Input = failed,

Output = [I, have, at, times]

CBOW

"**One day all** your hard work will pay off"

Input 1	Input 2	Output
One	all	day

"One **day all your** hard work will pay off"

Input 1	Input 2	Output
One	all	day
day	your	all

"One day **all your hard** work will pay off"

Input 1	Input 2	Output
One	all	day
day	your	all
all	hard	your

"One day all **your hard work** will pay off"

Input 1	Input 2	Output
One	all	day
day	your	all
all	hard	your
your	work	hard

Advantages of CBOW

- Being probabilistic is nature, it is supposed to perform superior to deterministic methods(generally).
- It is low on memory. It does not need to have huge RAM requirements like that of co-occurrence matrix where it needs to store three huge matrices.

Disadvantages of CBOW

- CBOW takes the average of the context of a word. For example, Apple can be both a fruit and a company but CBOW takes an average of both the contexts and places it in between a cluster for fruits and companies.
- Training a CBOW from scratch can take forever if not properly optimized.

Skip-Gram

"**One day all** your hard work will pay off"

Input 1	Output
day	One
day	all

"One **day all your** hard work will pay off"

Input 1	Output
day	One
day	all
all	day
all	your

“One day all **your hard work** will pay off”

Input 1	Output
day	One
day	all
all	day
all	your
your	all
your	hard
hard	your
hard	work

Advantages of Skip Gram

- It is unsupervised learning hence can work on any raw text given.
- It requires less memory comparing with other words to vector representations.
- It requires two weight matrix of dimension [O, —v—] each instead of [—v—, —v—]. And usually, O is around 300 while —v— is in millions. So, we can see the advantage of using this algorithm.

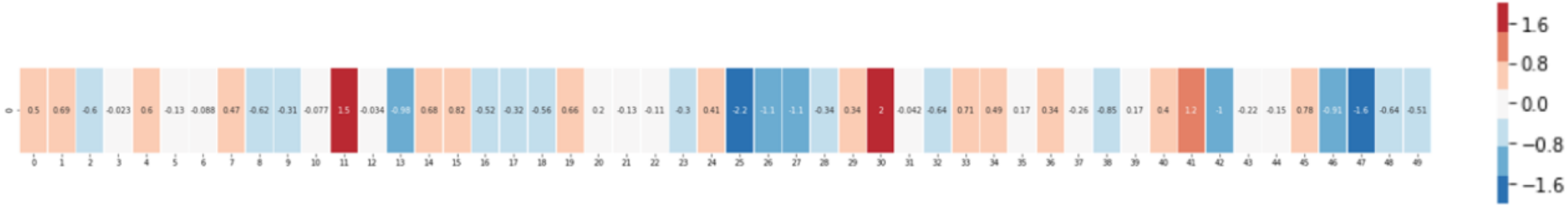
Disadvantages of Skip Gram

- Finding the best value for O and w is difficult.
- The time required for training this algorithm is high.

"CBOW and skip-gram cannot capture the polysemy, because they tend to represent a word as a single vector"

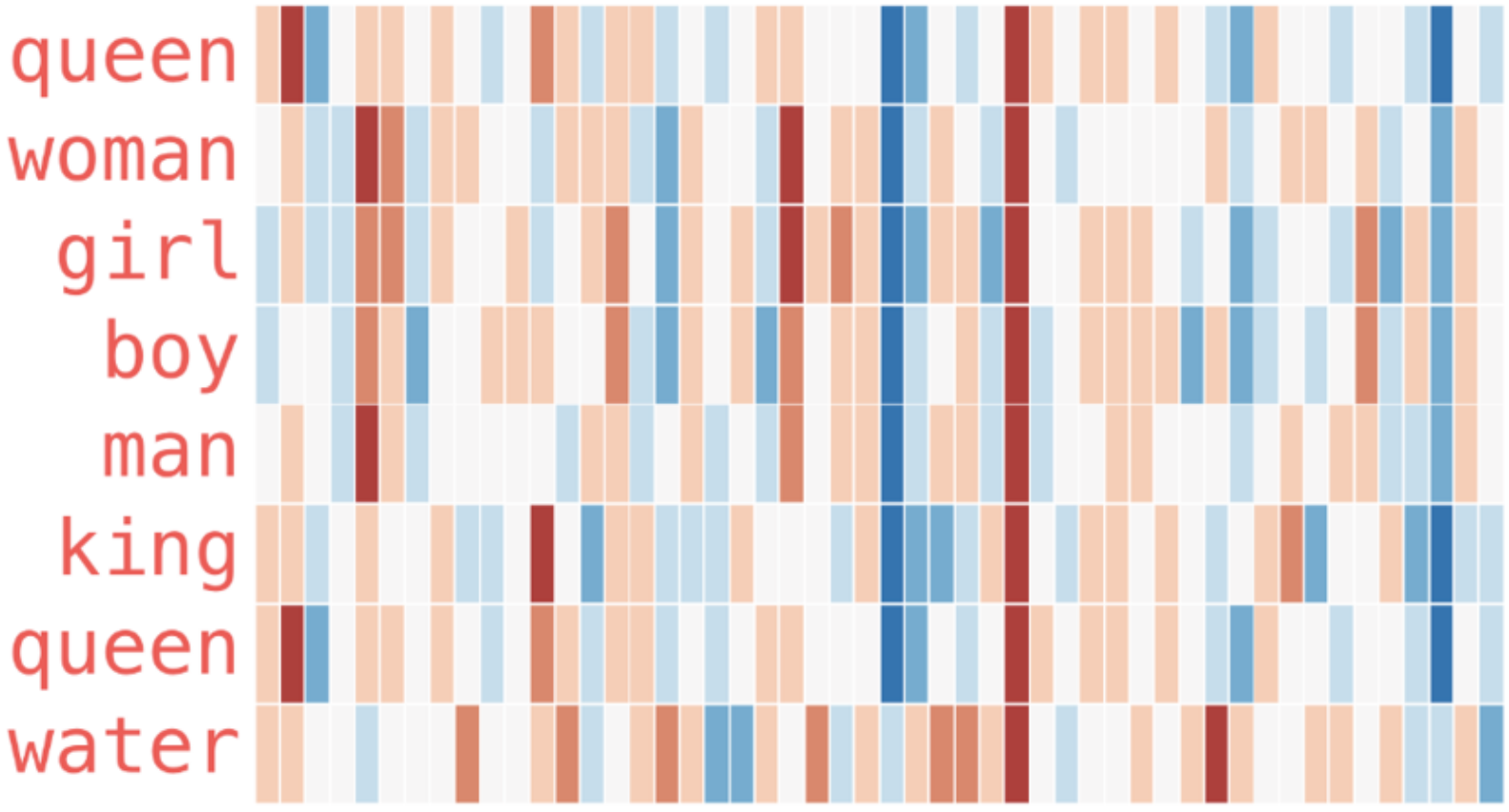
Word Analogies Example

Let below is the word embedding for word “King”



Let’s color code the cells based on their values (red if they’re close to 2, white if they’re close to 0, blue if they’re close to -2)

Word Analogies Example



https://jalammar.github.io/illustrated-word2vec/

Word Analogies Example: Observations

- There's a straight red column through all of these different words. They're similar along that dimension (and we don't know what each dimensions codes for)
- You can see how "woman" and "girl" are similar to each other in a lot of places. The same with "man" and "boy"
- "boy" and "girl" also have places where they are similar to each other, but different from "woman" or "man". These could be coding for a vague conception of youth.
- There are clear places where "king" and "queen" are similar to each other and distinct from all the others. These could be coding for a vague concept of royalty.
- All but the last word are words representing people. see that blue column going all the way down and stopping before the embedding for "water". This shows the differences between categories.

Word Analogies Example

king − man + woman ~= queen

- Fernández, Raquel. ”The Oxford Handbook of Computational Linguistics 2nd edition.”
- Goldberg, Yoav. ”Neural network methods for natural language processing.” Synthesis lectures on human language technologies 10.1 (2017): 1-309.
- Loughran, Tim, and Bill McDonald. ”When is a liability not a liability? Textual analysis, dictionaries, and 10-Ks.” The Journal of finance 66.1 (2011): 35-65.
- Brown, Tom, et al. ”Language models are few-shot learners.” Advances in neural information processing systems 33 (2020): 1877-1901.