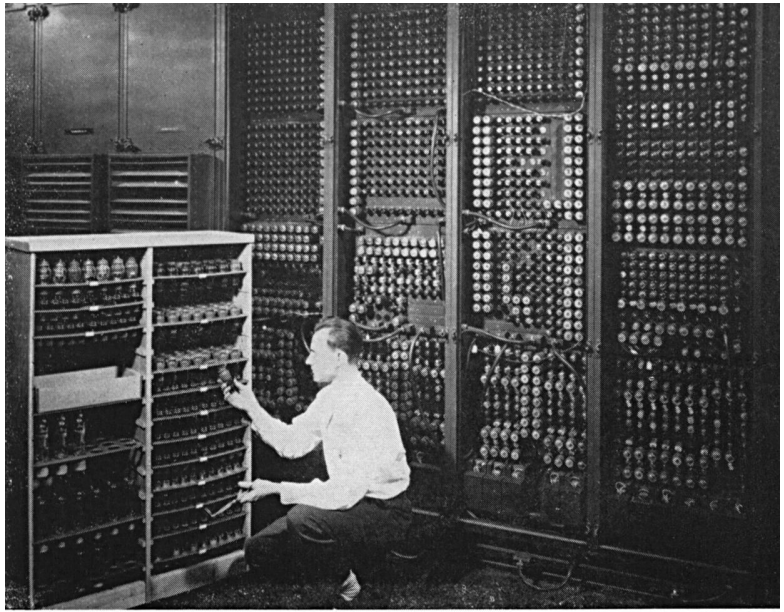
Natural Language Processing

Arin Ghazarian
Chapman University

Natural Language Processing

- Multidisciplinary field related to linguistics, computer science, artificial intelligence, and human-computer interaction (HCI).
- It emerged around 1950
- Breakthrough in NLP in the recent years due to
  - fast computer hardware like GPUs (graphics processing units),
  - Techniques like deep learning.

https://en.wikipedia.org/wiki/Computer#/media/File:ENIAC-changing_a_tube.jpg

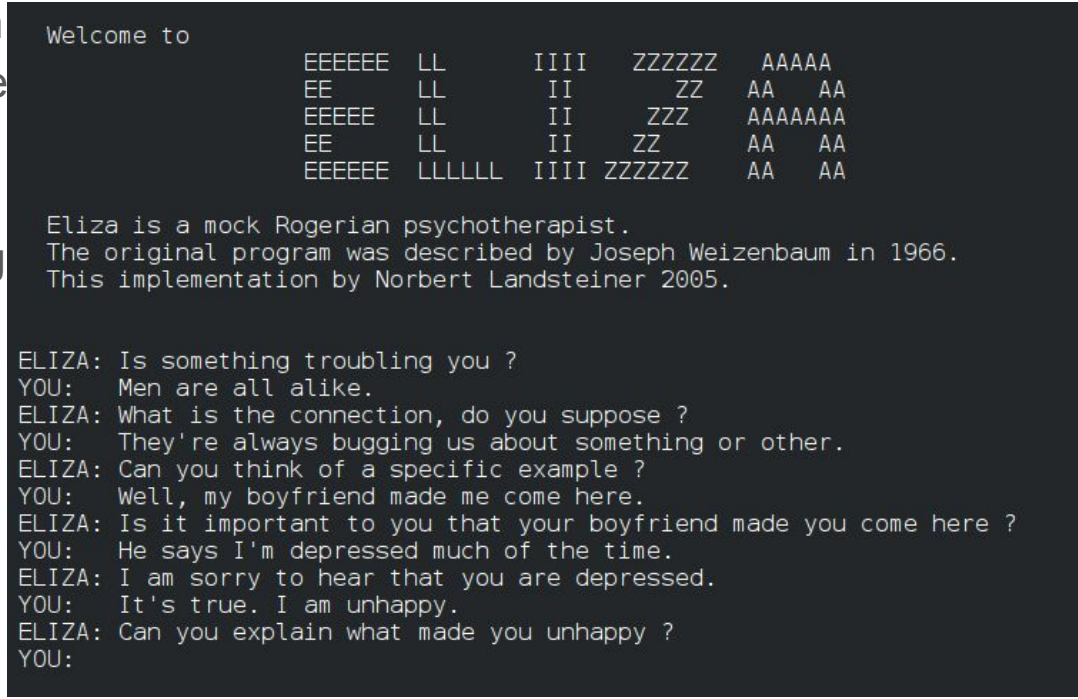


Replacing a bad tube meant checking among ENIAC's 19,000 possibilities.

ELIZA (1964-1966)

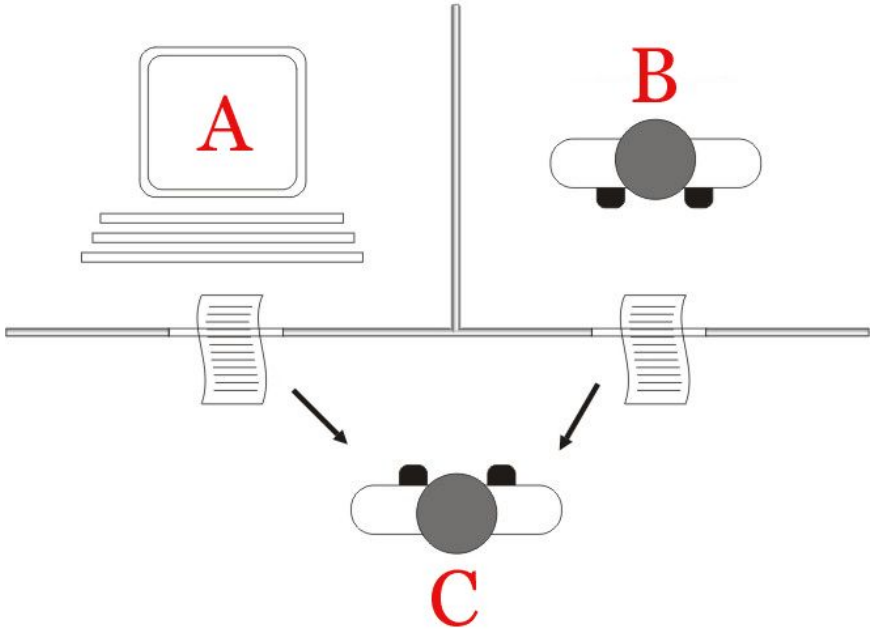
https://en.wikipedia.org/wiki/ELIZA

- ELIZA was an early chatbot system that convinced users that they were talking to an actual human
- DOCTOR, simulated a psychotherapist by simply parroting back at patients what they had just said



Turing Test

- Criterion of intelligence
  - Can a machine exhibit intelligent behavior indistinguishable from that of a human
- Is it possible to distinguish between human-to-human and human-to-machine natural language conversations?



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

NLP

- Processing and understanding natural language content such as speech and text.
- Speech recognition, natural language understanding (NLU), or natural language generation (NLG)

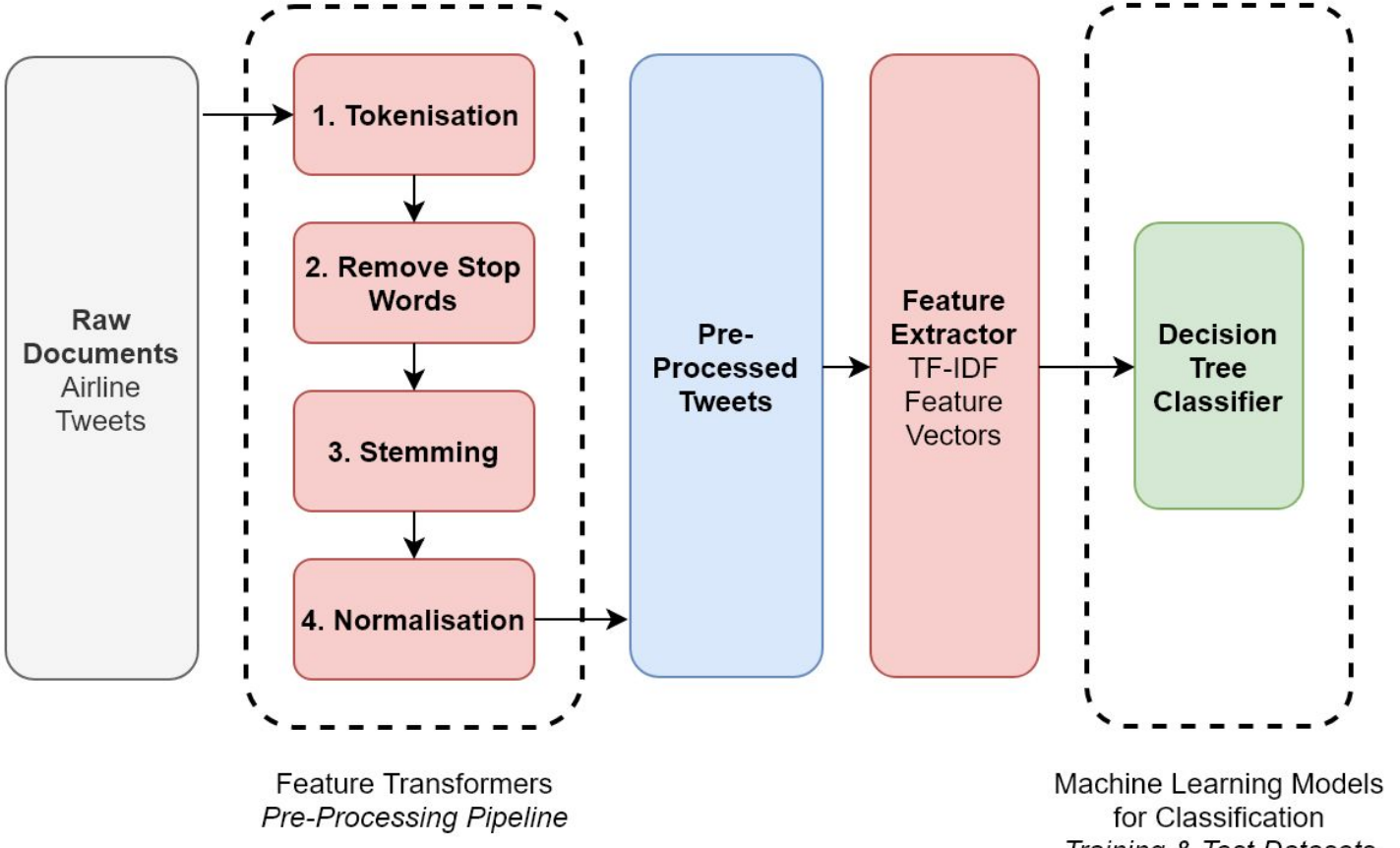
Applications

- Sentiment analysis:
- Text summarization
- Book generation:
  - In 2018, the novel 1 the Road was published as the first work by a neural network (Merchant, B., 2018).
- Machine translation
- Answering questions
- Fake news detection
- Chatterbots
- Clinical text mining
  - applying NLP on patient history documents and assessments to extract insights

NLP Pipeline

https://www.oreilly.com/library/view/machine-learning-with/9781789346565/def879fb-6164-4f84-992b-d61f5f7d5a83.xhtml

- Classical NLP systems had an architecture like this:



Text Preprocessing

Word Tokenization

- Break this sentence down into units like words

The leaves of these trees are green during the spring and summer.

[‘The’, ‘leaves’, ‘of’, ‘these’, ‘trees’, ‘are’, ‘green’, ‘during’, ‘the’, ‘spring’, ‘and’, ‘summer’]

Stop Word Removal

- A stop word is a commonly used word such as “the”, “a”, “is”, “are” that we can ignore and remove because these words are not useful for our task.
- There is no universal or common list of “stop words” used by all NLP tools.
- Stop words might differ from one application to another

[‘leaves’,‘trees’, ‘green’, ‘during’, ‘spring’, ‘summer’]

Stemming and Lemmatization

- Replace each word with its root
- Converting words to their canonical form
- Stemming: removing the suffix (the result might not be a dictionary word):

[‘leav’,‘tree’, ‘green’, ‘during’, ‘spring’, ‘summer’]

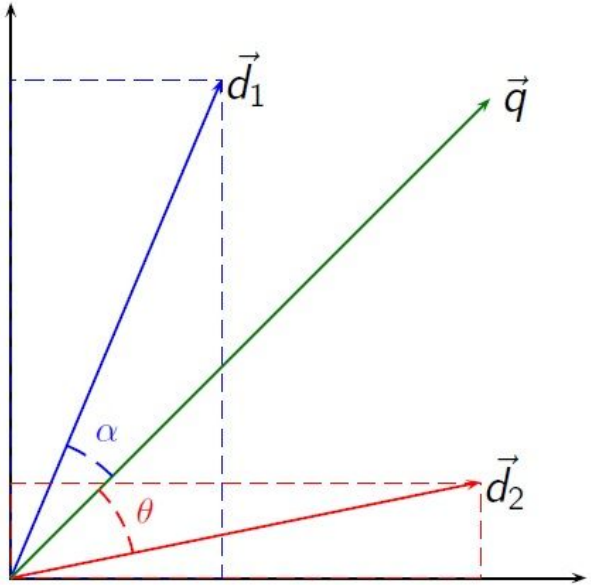
- leav is not a valid word and doesn’t exist in dictionaries. we can apply lemmatization if we need valid root words:

[‘leaf’,‘tree’, ‘green’, ‘during’, ‘spring’, ‘summer’]

Vectorization

Vector Spaces

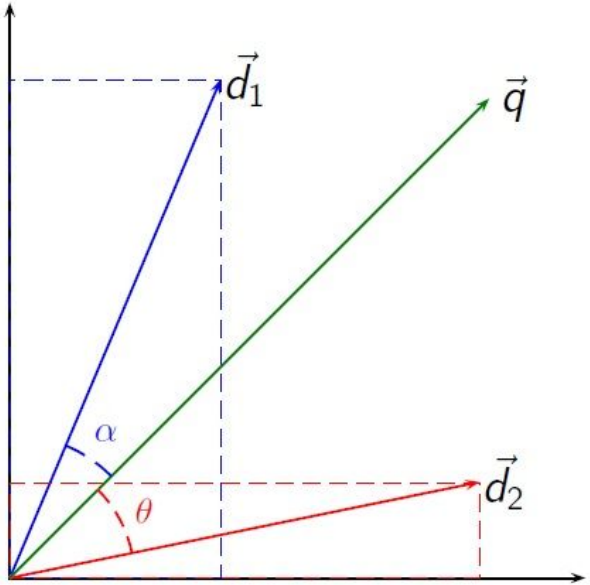
- Computers can only understand numbers and digits
- We can’t feed words into machine learning algorithms directly and we have to need to convert words into numbers by a process called vectorization.



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

Vector spaces

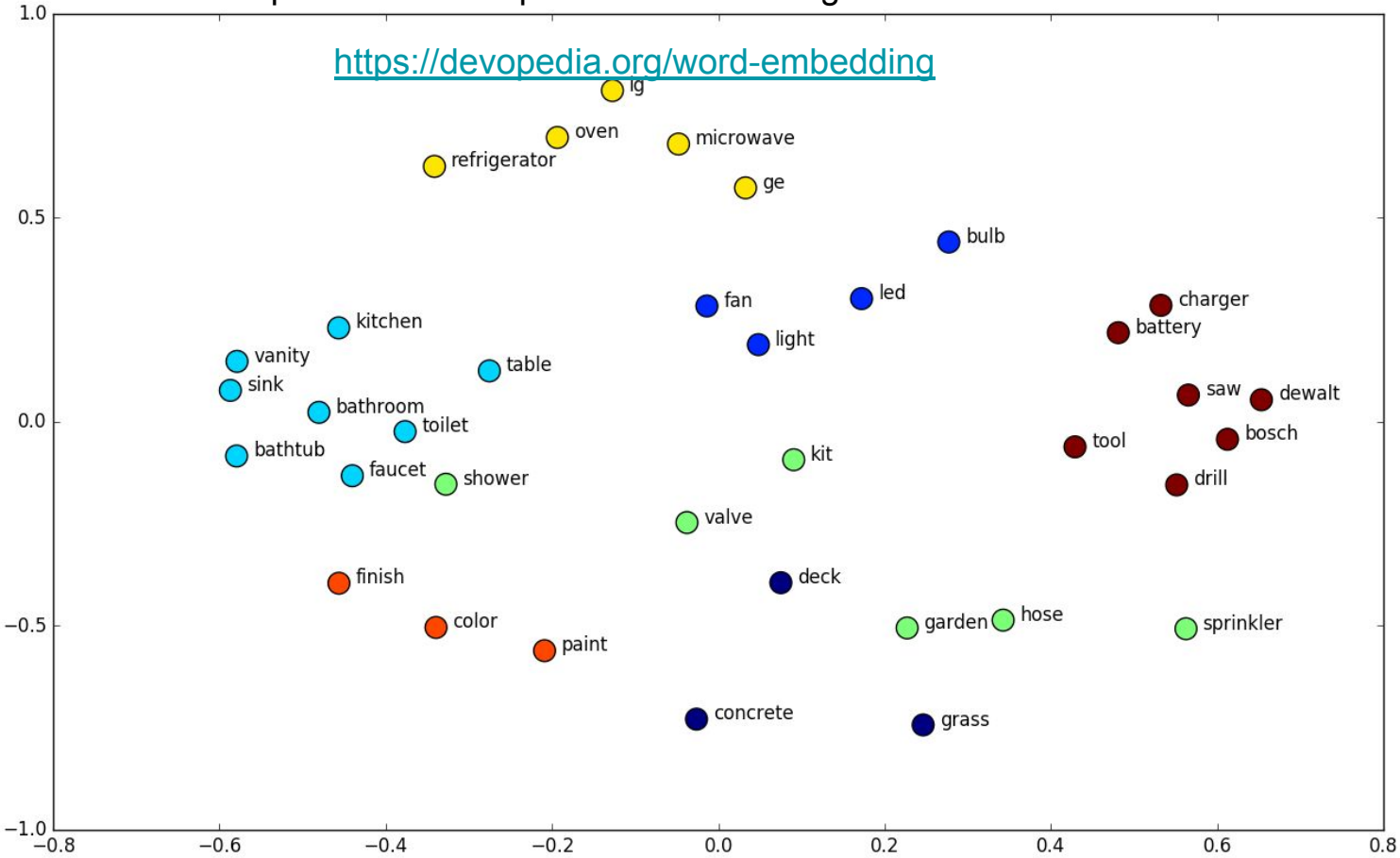
- the cosine of the angle two vectors=similarity between the two vector.
- Figure below shows a text search query from a collection of documents. Relevant documents vectors have a smaller angle with the query vector



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

$$cos(d_j, q) = \frac{d_j \cdot q}{\|d_j\| \|q\|} = \frac{\sum_{i=1}^{N} w_{i,j} w_{i,q}}{\sqrt{\sum_{i=1}^{N} w_{i,j}^2} \sqrt{\sum_{i=1}^{N} w_{i,q}^2}}$$

Example of a vector space model showing a cluster of related words



One-Hot Encoding

- The most basic text vectorization technique
- Each word is one dimension in the vector
- If that word appears in the text, the corresponding value =1 else =0

Corresponding word	One-hot encoding vector
spring	**1**
summer	**1**
during	**1**
green	**1**
like	**0**
my	**0**
tree	**1**
cat	**0**
dog	**0**
red	**0**
leaf	**1**
…	**0**

Bag-of-Words

- similar to one-hot encoding (each row in the vector corresponds to one word)
- In contrast to One-hot encoding, it also considers the number of times a word appears
- **Problem**:
  - high count for a word != word importance.
  - For instance, common words like “want” and “is” have high frequency, but that doesn't indicate importance.

Corresponding word	One-hot encoding vector
spring	**2**
summer	**10**
during	**3**
green	**1**
like	**4**
my	**0**
tree	**2**
cat	**0**
dog	**3**
red	**0**
leaf	**1**
…	**0**

TF-IDF (Term Frequency, Inverse Document Frequency)

- Normalize the term frequencies by multiplying it by the inverse of document frequency (This addresses the count=importance problem in BoW model)
- The tf–idf value is offset by the number of documents in the entire corpus that contain the word.

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

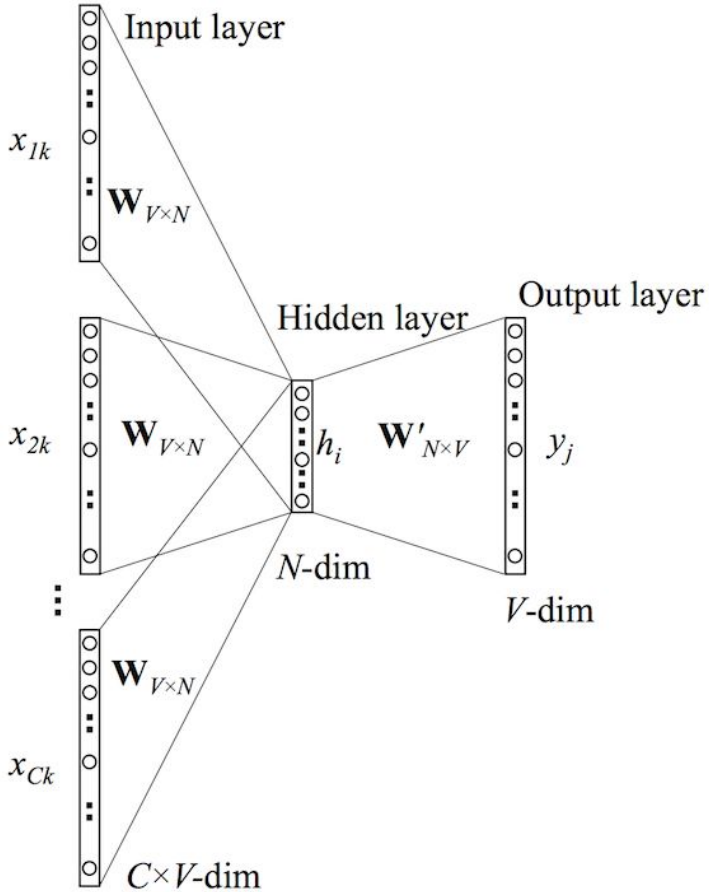
$$\mathrm{IDF}(t,D) = \log(N/n_t)$$

Word Embeddings

- One-hot encoding, BoW, TF-IDF disregards the grammar or ordering of the words
- Word embedding are real-valued vectors that encode the meaning of words.
- Words that have similar meaning are closer in the vector space.
- Lower dimensional and dense space
  - In previous methods the size of vector was equal to the size of dictionary
- Since it converts the original space with many dimensions per word to a lower dimensional, dense, and continuous vector space.
- Word embeddings consider the context of a word (i.e., other words that accompany a word)

Word2Vec

- uses shallow neural networks with one hidden layer and one output layer.
- For example, one neural network architecture to generate Word2vec embeddings is called CBOW (continuous bag-of-words) as shown here.



Word2Vec

- input: the one-hot encoding of words surrounding a given word (x1k, x2k,..., xCk).
  - Each surrounding word is then converted to a V-dimensional one-hot encoding vector(assuming there are V words in the dictionary)
  - Since we have C words, our input vector will be C×V elements.
- The expected output vector (yj) is a V-dimensional one-hot encoding for the target word.
- There is one hidden layer with N units
- Each input word vector is connected to this hidden layer with a weight matrix of W of V×N dimensions.
- the C N-dimensional vectors from the hidden layer will be averaged element-wise and the result will be the activation layer to be fed into the final softmax layer.
- After the training is done, the weights from the hidden layer can be used to generate a Word2vec vector of size N for any given word.

Word2Vec

https://towardsdatascience.com/word-embeddings-for-nlp-5b72991e01d4