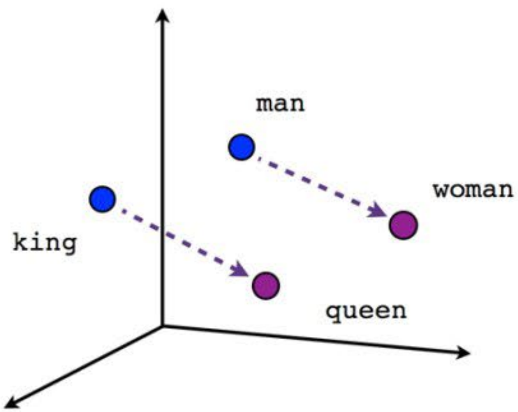
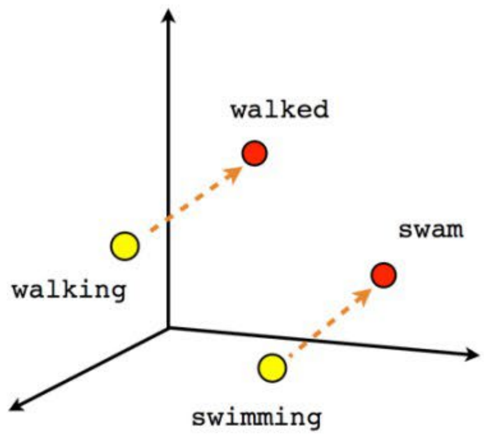
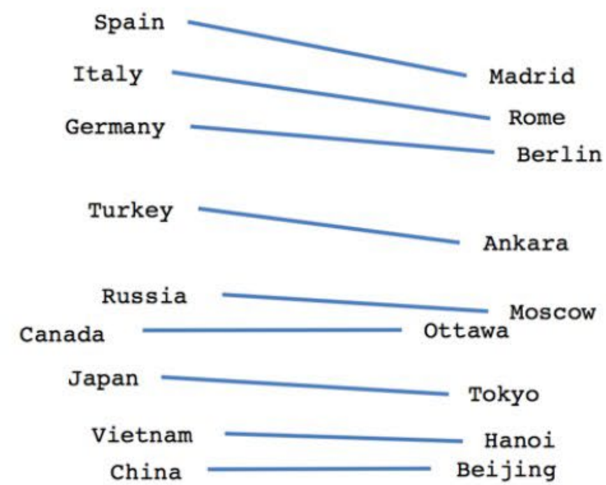
Part IV: Text Representations



Male-Female



Verb tense



Country-Capital

Instructor:
Evangelos Kalogerakis

Why do we need text representations?

Sentiment analysis in text & speech

"I hate artificial intelligence" => negative view (capture meaning of "hate")

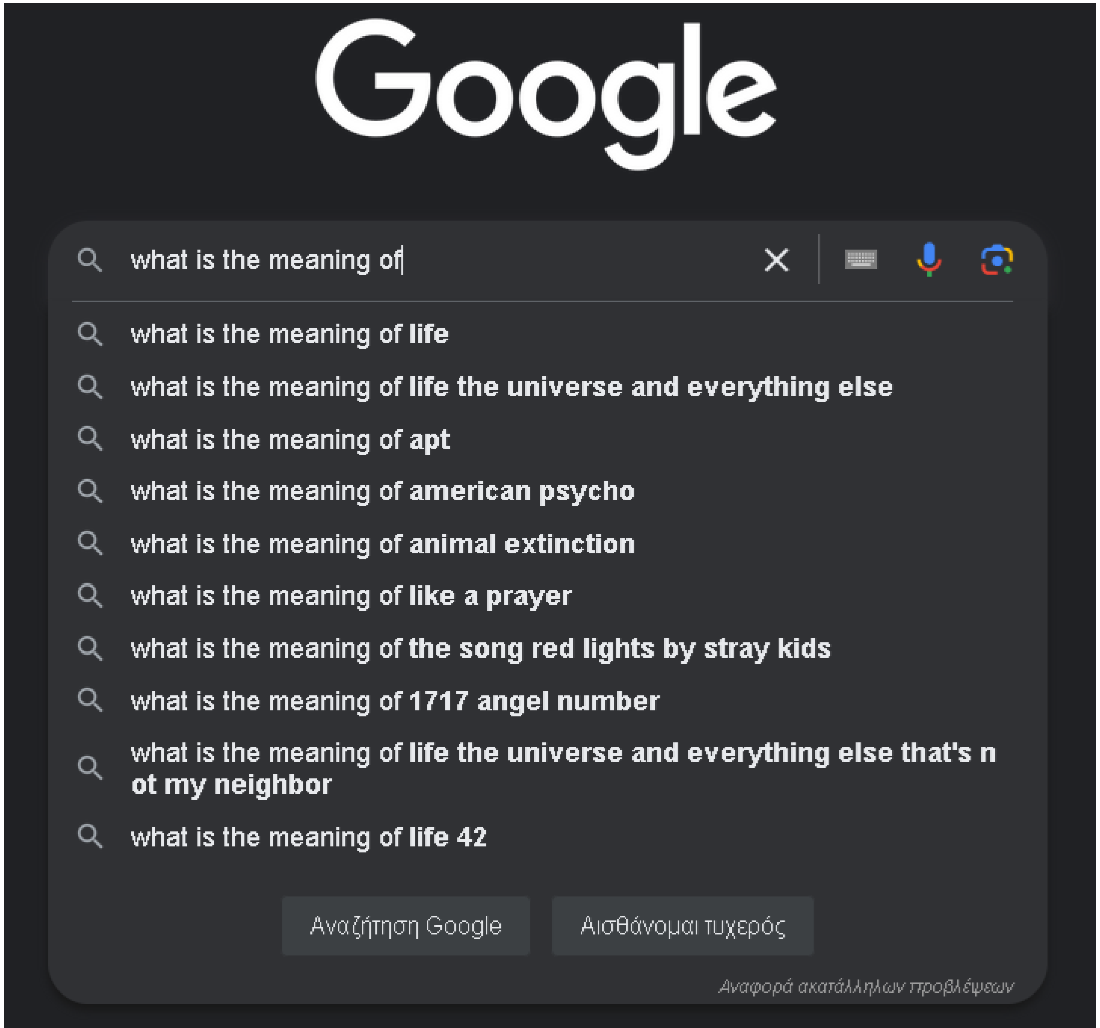
"Artificial intelligence is my thing" => positive view (capture meaning of "my thing")

"I do not like artificial intelligence" => negative view (must handle negation)

"Artificial intelligence is difficult, but I can tolerate it" => neutral view

Why do we need text representations?

Language models



Language models

Goal: compute the probability of a sentence or sequence of words:

$$P(\boldsymbol{W}) = P(Word_1, Word_2, Word_3, Word_4, Word_5, \ldots, Word_n)$$

Language models

Goal: compute the probability of a sentence or sequence of words:

$$P(\boldsymbol{W}) = P(Word_1, Word_2, Word_3, Word_4, Word_5, \ldots, Word_n)$$

Related task: probability of an upcoming word:

$$P(Word_5 \mid Word_1, Word_2, Word_3, Word_4)$$

Language models

Goal: compute the probability of a sentence or sequence of words:

$$P(\boldsymbol{W}) = P(Word_1, Word_2, Word_3, Word_4, Word_5, \ldots, Word_n)$$

Related task: probability of an upcoming word:

$$P(Word_5 \mid Word_1, Word_2, Word_3, Word_4)$$

A model that computes either of these:

$$P(\boldsymbol{W}) \quad \text{or} \quad P(Word_n \mid Word_1, Word_2, Word_3, \ldots, Word_{n-1})$$

is called a **language model or LM**

Eh? Probability?

Degree of confidence for an **event** to happen (or frequency of an event)

e.g. choose *"life"* as a word to use from a dictionary

Eh? Probability?

Degree of confidence for an **event** to happen (or frequency of an event)

e.g. choose *"life"* as a word to use from a dictionary

Event E is an **outcome (or a set of outcomes) from the space of all possible outcomes $\boldsymbol{\Omega}$**

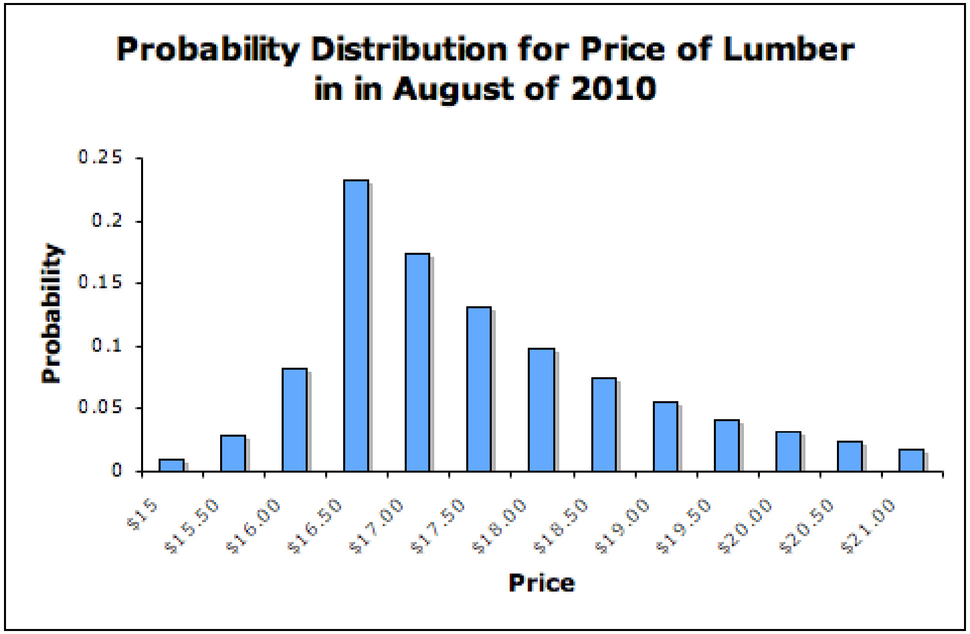
e.g. $\boldsymbol{\Omega} = \{..., \text{"}lieu\text{"}, \text{"}lieutenant\text{"}, \text{"}lieve\text{"}, \text{"}life\text{"}, \text{"}lifeboat\text{"}, ...\}$

The above event is $E = \{\text{"}life\text{"}\}$

What is a probability distribution?

A probability distribution defines a probability from events to real values.

- Probabilities are non-negative.
- Smaller than or equal to 1.



Review

Summary of probabilities

Event	Probability
A	$P(A) \in [0,1]$
not A	$P(A^{\complement}) = 1 - P(A)$
A or B	$P(A \cup B) = P(A) + P(B) - P(A \cap B)$ $P(A \cup B) = P(A) + P(B) \qquad \text{if A and B are mutually exclusive}$
A and B	$P(A \cap B) = P(A \mid B)P(B) = P(B \mid A)P(A)$ $P(A \cap B) = P(A)P(B) \qquad \text{if A and B are independent}$
A given B	$P(A \mid B) = \frac{P(A \cap B)}{P(B)} = \frac{P(B \mid A)P(A)}{P(B)}$

Conditional Probability

Suppose we want to reason about the next word in a sentence
e.g., *“what is the meaning of …”*

Conditional Probability

Suppose we want to reason about the next word in a sentence
e.g., *“what is the meaning of …”*

$P(\text{“life”})$ means the **probability of picking this word in general** e.g., we can measure the frequency of this word in a collection of documents

$$P(\text{“life”}) = count(\text{“life”}) \,/\, \#words$$

Conditional Probability

Suppose we want to reason about the next word in a sentence
e.g., *"what is the meaning of …"*

P("life") means the **probability of picking this word in general** e.g., we can measure the frequency of this word in a collection of documents

$P(\text{“life”}) = count(\text{“life”})\ /\ \#words$

However, **given that we know some words before** (i.e., an event that has already happened), the probability of picking *"life"* changes!

$$P(\text{“life”} \mid \text{“what is the meaning of”})$$

Conditional probability

Conditional Probability

$$P(A \mid B) = \frac{P(A,B)}{P(B)}$$

In the previous example, we can count frequencies:

$$P(A \mid B) = \frac{count(\text{“}what\ is\ the\ meaning\ of\ life\text{”})}{count(\text{“}what\ is\ the\ meaning\ of\text{”})}$$

Chain rule of probability

$$P(A,B) = P(A \mid B)\,P(B)$$

More variables:

$$P(A,B,C,D) \;=\; P(A)\,P(B \mid A)\,P(C \mid A,B)\,P(D \mid A,B,C)$$

Chain rule of probability

The Chain Rule applied to compute joint probability of words in sentence:

$$P(w_1\ w_2\ w_3\ \ldots\ w_n) = \prod_{i=1}^{n} P(w_i \mid w_1\ w_2\ w_3\ \ldots\ w_{i-1})$$

Chain rule of probability

The Chain Rule applied to compute joint probability of words in sentence:

$$P\left(w_1\ w_2\ w_3\ \dots\ w_n\right) = \prod_{i=1}^{n} P\left(w_i \,|\, w_1\ w_2\ w_3\ \dots\ w_{i-1}\right)$$

Example:

$$\begin{aligned} P\left(\text{"}what\ is\ the\ meaning\ of\ life\text{"}\right) = {} & P(\text{"}what\text{"}) \cdot P(\text{"}is\text{"} \,|\, \text{"}what\text{"}) \cdot P(\text{"}the\text{"} \,|\, \text{"}what\ is\text{"}) \\ & \cdot P(\text{"}meaning\text{"} \,|\, \text{"}what\ is\ the\text{"}) \cdot P(\text{"}of\text{"} \,|\, \text{"}what\ is\ the\ meaning\text{"}) \\ & \cdot P(\text{"}life\text{"} \,|\, \text{"}what\ is\ the\ meaning\ of\text{"}) \end{aligned}$$

Markov Assumption

Andrei Markov
(1856~1922)

Simplifying assumption:

$$P(\text{"}life\text{"}\,|\,\text{"}what\ is\ the\ meaning\ of\text{"}) \approx P(\text{"}life\text{"}\,|\,\text{"}the\ meaning\ of\text{"})$$

or

$$P(\text{"}life\text{"}\,|\,\text{"}what\ is\ the\ meaning\ of\text{"}) \approx P(\text{"}life\text{"}\,|\,\text{"}meaning\ of\text{"})$$

or

$$P(\text{"}life\text{"}\,|\,\text{"}what\ is\ the\ meaning\ of\text{"}) \approx P(\text{"}life\text{"}\,|\,\text{"}of\text{"})$$

Markov Assumption

Andrei Markov
(1856~1922)

Simplifying assumption:

condition on up to k previous words

$$P\left(w_i \mid w_1\ w_2\ w_3\ \ldots\ w_{i-1}\right) \approx P\left(w_i \mid w_{i-k}\ w_{i-k+1} \ldots\ w_{i-1}\right)$$

Unigram model

Andrei Markov (1856~1922)

Oversimplification:

$$P(w_1\ w_2\ w_3\ \ldots\ w_n) \approx \prod_{i=1}^{n} P(w_i)$$

Some automatically generated sentences from a unigram LM:

fifth, an, of, futures, the, an, incorporated, a, a, the, inflation, most, dollars, quarter, in, is, mass, thrift, did, eighty, said, hard, 'm', july, bullish, that, or, limited, the

Bigram (2-gram) model

Andrei Markov (1856~1922)

Less simplification:

$$P(w_1\ w_2\ w_3\ \ldots\ w_n) \approx \prod_{i=1}^{n} P(w_i \mid w_{i-1})$$

$$P(w_i \mid w_{i-1}) = \frac{count(w_{i-1}, w_i)}{count(w_{i-1})}$$

Approximating Shakespeare

1 gram	–To him swallowed confess hear both. Which. Of save on trail for are ay device and rote life have –Hill he late speaks; or! a more to leg less first you enter
2 gram	–Why dost stand forth thy canopy, forsooth; he is this palpable hit the King Henry. Live king. Follow. –What means, sir. I confess she? then all sorts, he is trim, captain.
3 gram	–Fly, and will rid me these news of price. Therefore the sadness of parting, as they say, 'tis done. –This shall forbid it should be branded, if renown made it empty.
4 gram	–King Henry. What! I will go seek the traitor Gloucester. Exeunt some of the watch. A great banquet serv'd in; –It cannot be but so.

… generated sentences get "better" as we increase the model order

The problem of "zero" frequencies

Training set:

... denied the allegations

... denied the reports

... denied the claims

... denied the request

$P(\text{"offer"} \mid \text{denied the}) = 0$ **?**

Smoothing

Before smoothing:

P(? | denied the)

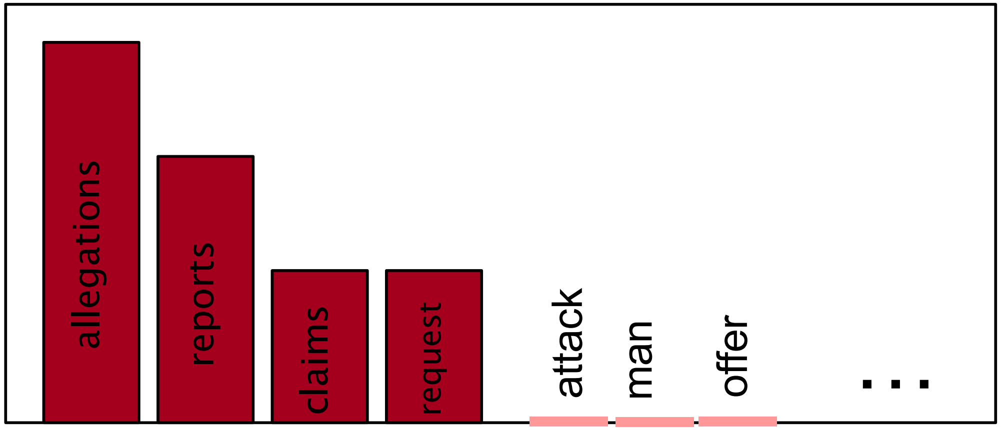
3 allegations

2 reports

1 claims

1 request

7 total



Add pseudo-counts e.g.,:

5 allegations

4 reports

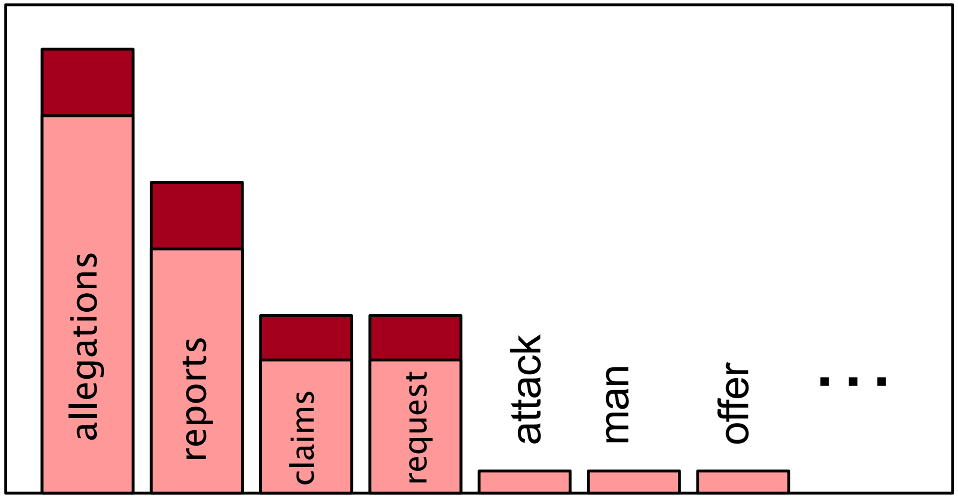
3 claims

3 request

2 attack

2 offer

…



Problems…

Smoothing promotes unlikely completions… e.g., “denied the unmelancholy”

Problems…

Smoothing promotes unlikely completions… e.g., “denied the unmelancholy”

Huge storage space to store long n-grams

Problems…

Smoothing promotes unlikely completions… e.g., “denied the unmelancholy”

Huge storage space to store long n-grams

We treat all words / prefixes independently of each other!

students opened their ___

pupils opened their ___

scholars opened their ___

undergraduates opened their ___

students turned the pages of their ___

students attentively perused their ___

Shouldn’t we *share information* across these semantically-similar prefixes?

Representation problem

We do not have any notion of word “meaning” (e.g., word synonyms, similarity, relatedness etc)

Representation problem

We do not have any notion of word “meaning” (e.g., word synonyms, similarity, relatedness etc)

We basically represented each word/n-gram as a vector of zeros with a single 1 identifying its index in the vocabulary

Representation problem

We do not have any notion of word "meaning" (e.g., word synonyms, similarity, relatedness etc)

We basically represented each word/n-gram as a vector of zeros with a single 1 identifying its index in the vocabulary

vocabulary
i
hate
love
the
movie
film

movie = [0, 0, 0, 0, 1, 0]

film = [0, 0, 0, 0, 0, 1]

what are the issues of representing a word this way?

Representation problem

movie = [0, 0, 0, 0, 1, 0]

film   = [0, 0, 0, 0, 0, 1]

…

Evaluate similarity?

Euclidean distance is the same between all pairs of words

Dot product is 0 – all vectors are orthogonal

Representation problem

movie = [0, 0, 0, 0, 1, 0]

film = [0, 0, 0, 0, 0, 1]

…

Evaluate similarity?

Euclidean distance is the same between all pairs of words

Dot product is 0 – all vectors are orthogonal

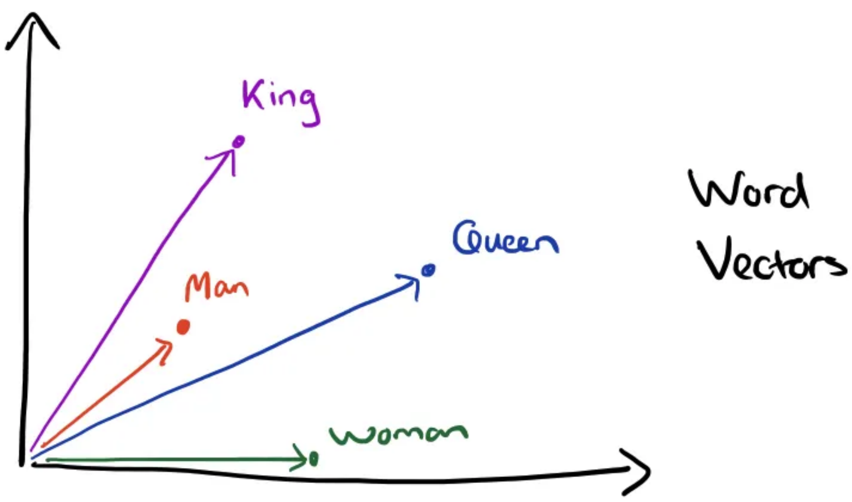
What we want is a representation space in which words, phrases, sentences etc. that are semantically similar also have similar representations!

Words as “embeddings”

Represent words with vectors called “**embeddings**” (Mikolov et al., NIPS 2013, word2vec)

e.g.,

king = [0.99, 0.99, 0.05, 0.7, -1 ….]
(this can have lots of dimensions)



The vectors for the words “King” and “Man” may be similar, as might the vectors for “Queen” and “Woman.

https://medium.com/@saschametzger/what-are-tokens-vectors-and-embeddings-how-do-you-create-them-e2a3e698e037

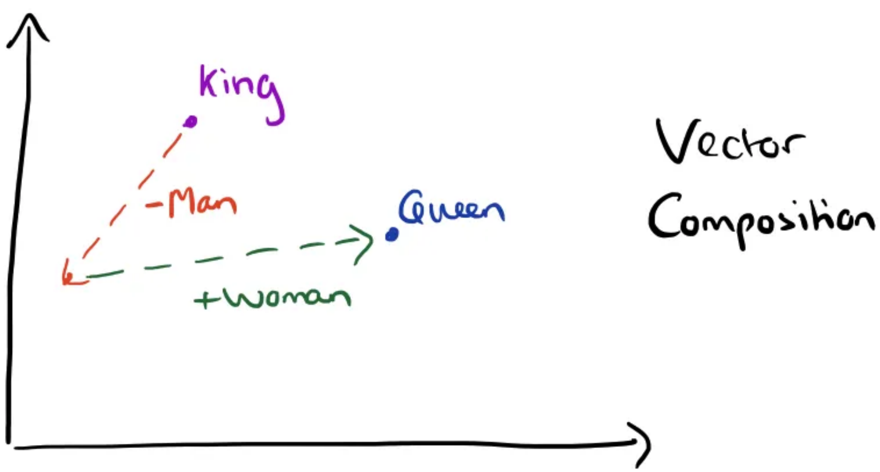
Words as “embeddings”

Represent words with vectors called “**embeddings**” (Mikolov et al., NIPS 2013, word2vec)

e.g.,

king = [0.99, 0.99, 0.05, 0.7, -1 ….]

(this can have lots of dimensions)



you may subtract the vector for “Man” from the vector for “King” and add the vector for “Woman” to get the vector for “Queen.”

https://medium.com/@saschametzger/what-are-tokens-vectors-and-embeddings-how-do-you-create-them-e2a3e698e037

Words as “embeddings”

Ideally, each entry encodes something about the word

It would be extremely hard to set these values by hand for each word!!!

https://medium.com/@saschametzger/what-are-tokens-vectors-and-embeddings-how-do-you-create-them-e2a3e698e037

What modern LMs (LLMs) do

[Step 1] Tokenization: Input text is split into "pieces", called **tokens** (e.g., words)

e.g.:

"The cat went up the stairs." =>

["The", " cat", " went", " up", " the", " stairs", "."]

Store as token "IDs":

[20, 4758, 439, 62, 5, 16745, 4]

(where id is a unique identifier for each token – an index in a list)

=> Word tokenization can result in a very large vocabulary!

https://medium.com/@saschametzger/what-are-tokens-vectors-and-embeddings-how-do-you-create-them-e2a3e698e037

What modern LMs (LLMs) do

[Step 1] Alternative tokenization: Input text is split into characters

e.g.:
"Hello world!"=>
["H", "e", "l", "l", "o", " ", "w", "o", "r", "l", "d", "!"]
Store them as token IDs

=> Mapping such tokens to useful embeddings is hard!

https://medium.com/@saschametzger/what-are-tokens-vectors-and-embeddings-how-do-you-create-them-e2a3e698e037

What modern LMs (LLMs) do

[Step 1] Yet another tokenization: Input text is split into subwords (morphemes -- the smallest meaningful units of language) -- they can also be as short as individual characters or complete words, depending on their frequency of occurrence.

e.g.:

"This is an example of the bert tokenizer!" =>

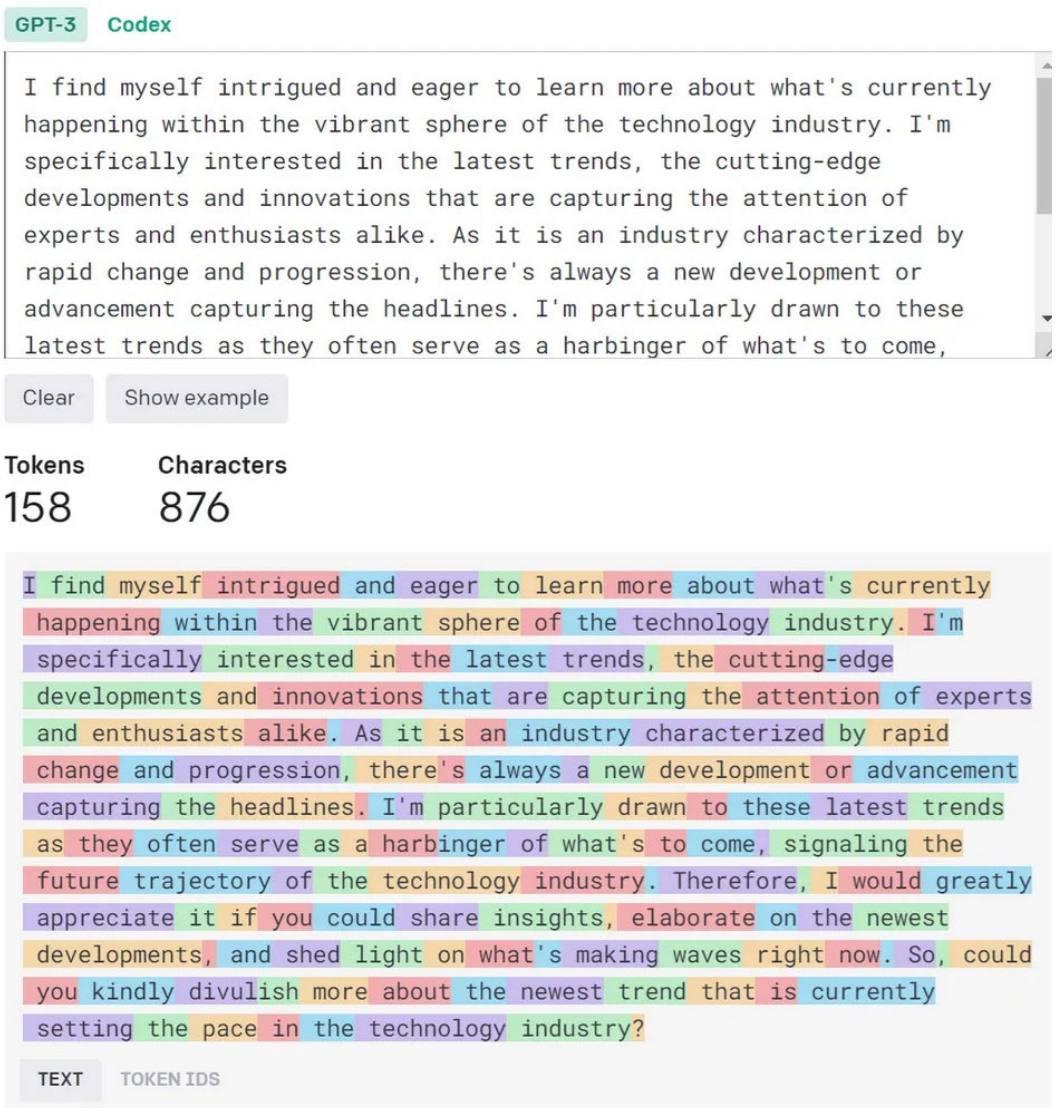
"This", " is", " an", " example", " of", " the", "bert", "token", "#izer", "!"]

Store them as token IDs

=> Best of both worlds!

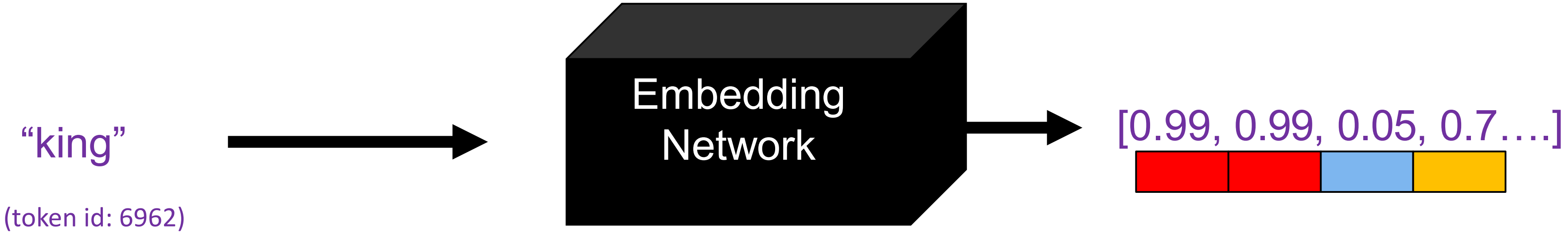
https://medium.com/@saschametzger/what-are-tokens-vectors-and-embeddings-how-do-you-create-them-e2a3e698e037

https://platform.openai.com/tokenizer



What modern LMs (LLMs) do

[Step 2] Embedding: token ids are converted into vectors



What modern LMs (LLMs) do

[Step 2] Embedding: token ids are converted into vectors

```
Token String     Token ID        Embedded Token Vector
       '<s>' ->    0 -> [ 0.1150, -0.1438,  0.0555, ... ]
     '<pad>' ->    1 -> [ 0.1149, -0.1438,  0.0547, ... ]
      '</s>' ->    2 -> [ 0.0010, -0.0922,  0.1025, ... ]
     '<unk>' ->    3 -> [ 0.1149, -0.1439,  0.0548, ... ]
         '.' ->    4 -> [-0.0651, -0.0622, -0.0002, ... ]
      ' the' ->    5 -> [-0.0340,  0.0068, -0.0844, ... ]
         ',' ->    6 -> [ 0.0483, -0.0214, -0.0927, ... ]
       ' to' ->    7 -> [-0.0439,  0.0201,  0.0189, ... ]
      ' and' ->    8 -> [ 0.0523, -0.0208, -0.0254, ... ]
       ' of' ->    9 -> [-0.0732,  0.0070, -0.0286, ... ]
        ' a' ->   10 -> [-0.0194,  0.0302, -0.0838, ... ]
                                ...
```

https://www.lesswrong.com/posts/pHPmMGEMYefk9jLeh/llm-basics-embedding-spaces-transformer-token-vectors-are

What modern LMs (LLMs) do

[Step 2] Embedding: create a **one-hot vector** $\boldsymbol{v}$ whose size is the size of the vocabulary of tokens. Make it **1** at the index of input token, and **0** at all other indices.

An embedding matrix $\mathbf{W_e}$ of size (#dimensions x #tokens) transforms this one-hot vector to the embedding as: $\boldsymbol{q} = \boldsymbol{W_e}\,\boldsymbol{v}$

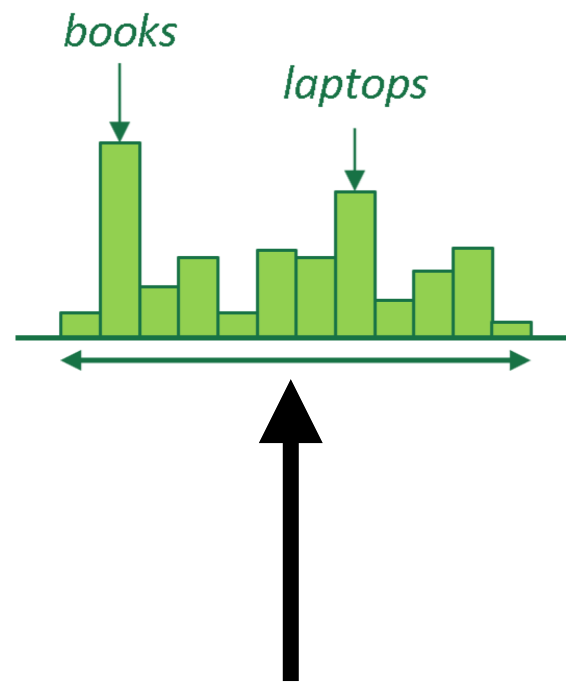
```
                    [ 0 ]           0 * [ 0.1150, -0.1438,  0.0555, ... ]
                    [ 0 ]         + 0 * [ 0.1149, -0.1438,  0.0547, ... ]
                    [ 0 ]         + 0 * [ 0.0010, -0.0922,  0.1025, ... ]
                    [ 0 ]  W      + 0 * [ 0.1149, -0.1439,  0.0548, ... ]
                    [ 0 ]   E     + 0 * [-0.0651, -0.0622, -0.0002, ... ]
' the' -> 5 ->      [ 1 ]  ->     + 1 * [-0.0340,  0.0068, -0.0844, ... ]
                    [ 0 ]         + 0 * [ 0.0483, -0.0214, -0.0927, ... ]
                    [ 0 ]         + 0 * [-0.0439,  0.0201,  0.0189, ... ]
                    [...]         + ...
```

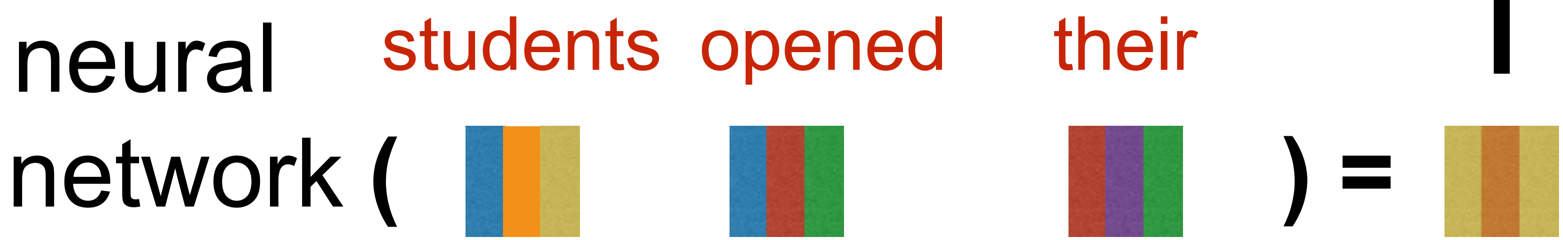
https://www.lesswrong.com/posts/pHPmMGEMYefk9jLeh/llm-basics-embedding-spaces-transformer-token-vectors-are

What modern LMs (LLMs) do

[Step 3] Neural network processing:

transform a sequence of vector representations into a probability distribution over the vocabulary to predict the next word (or a sentiment in sentiment analysis…)





NEXT TIME: NEURAL NETWORKS!