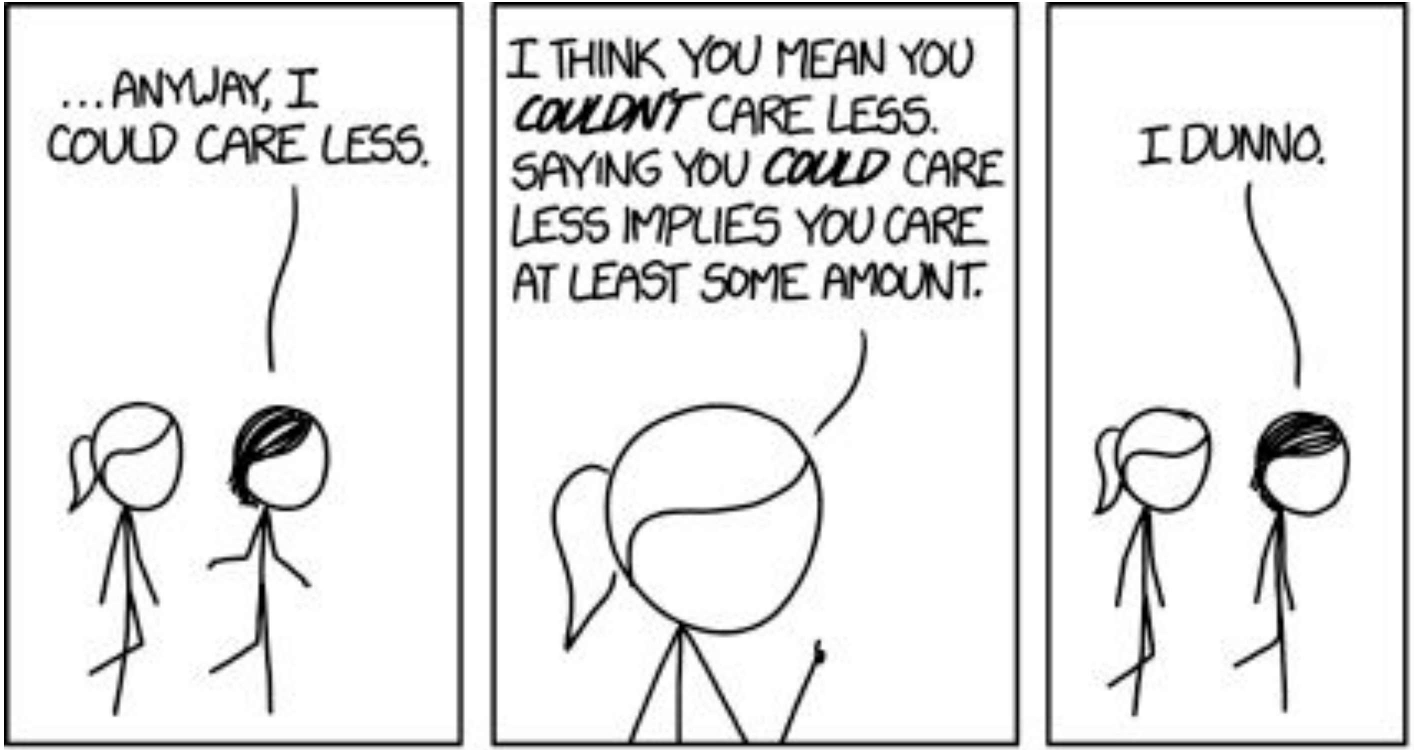
Foundations of Artificial Intelligence in Business
- Intro to NLP

Pearl Yu

Learning Goals

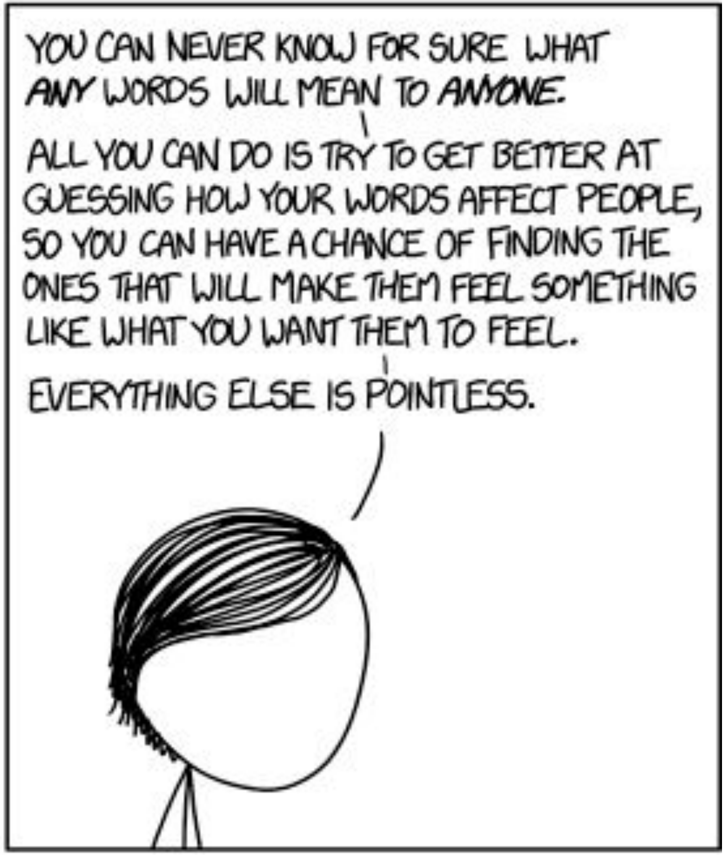
- *Simple business use cases.*
- A big picture understanding of human languages and the difficulties in understanding and producing them.
- Understand how to represent words and how representations are learnt via Word2Vec.
- *Architectures suitable for language inputs: RNN, transformers (LLM)*

Human Languages



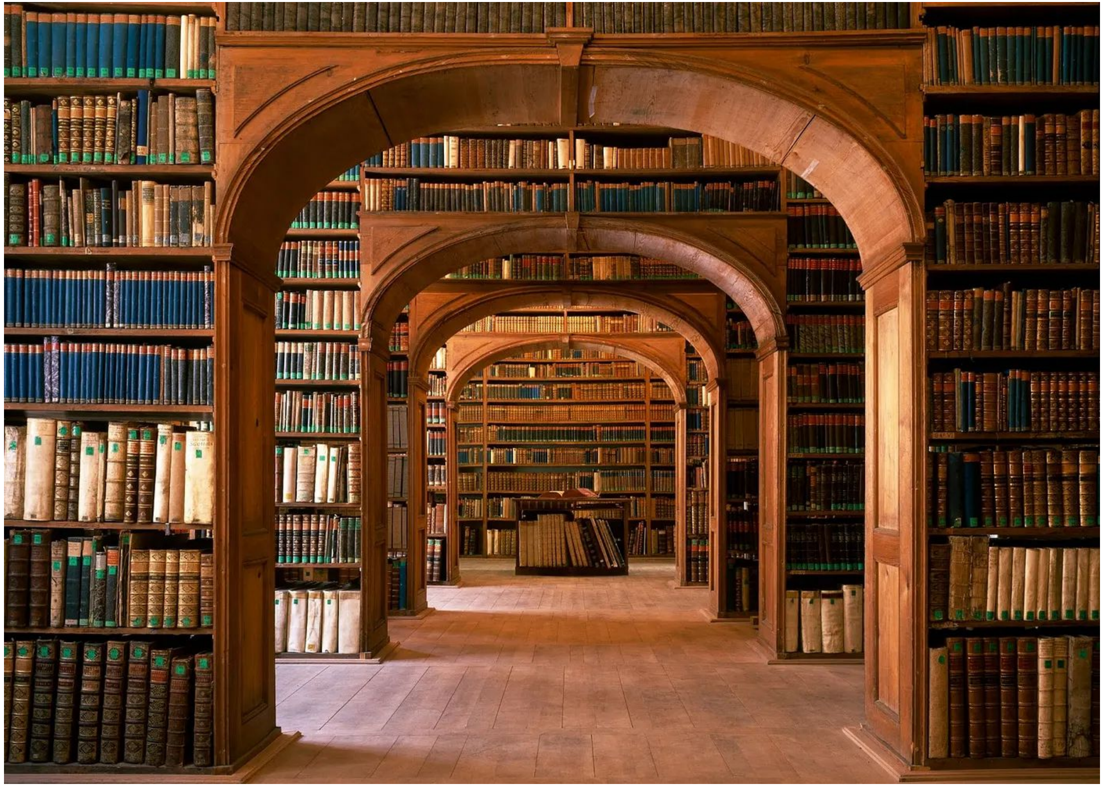
Human Languages

With Artificial Intelligence



Language Models

- Knowledge contained in language.



Language Models

- Knowledge contained in language.

One simple sentence but packs in a lot of stuff.

That there is a theory of relativity.

‘Einstein developed the theory of relativity.’

Who Einstein is? (a famous physicist).

That theories are things humans can create and share.

Implicitly, that this is important enough for us to mention in class and exams.”

Language Models

- Knowledge contained in language.

Ambiguity

‘My cat has nine lives.’

English speaker: Yeah it’s a metaphor. Telling a story?

Computer: You’re picturing a terrifying science experiment.

Language Models

- Knowledge contained in language.

Culture packed into words

‘That movie was fire!’

Not calling the fire department. This means awesome.

Language is social agreement. The meaning changes overtime too.

Language Models

- Knowledge contained in language.

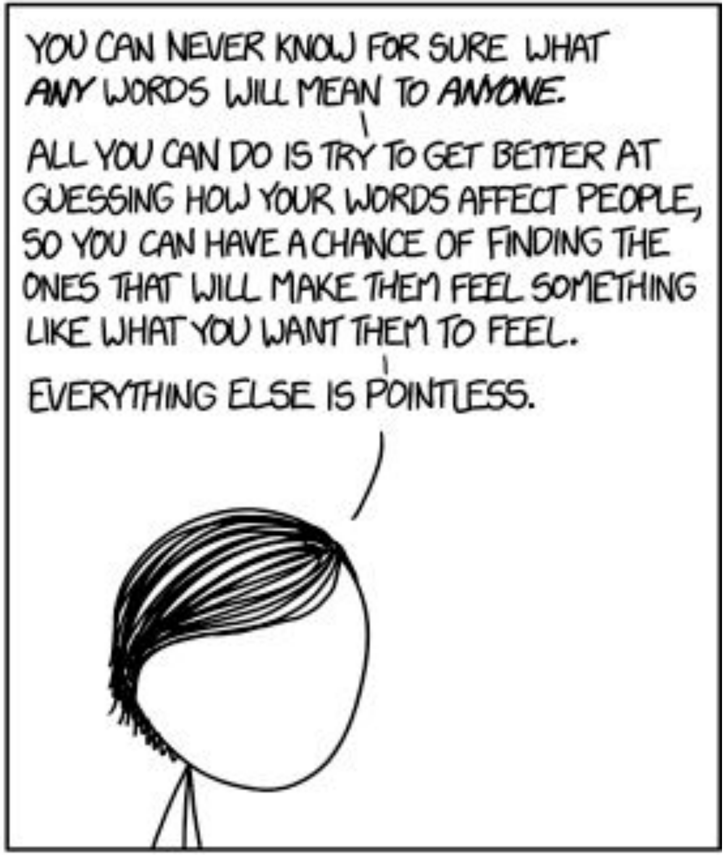
Implicit knowledge

‘Let’s meet at Starbucks.’

No need to explain what starbucks is.

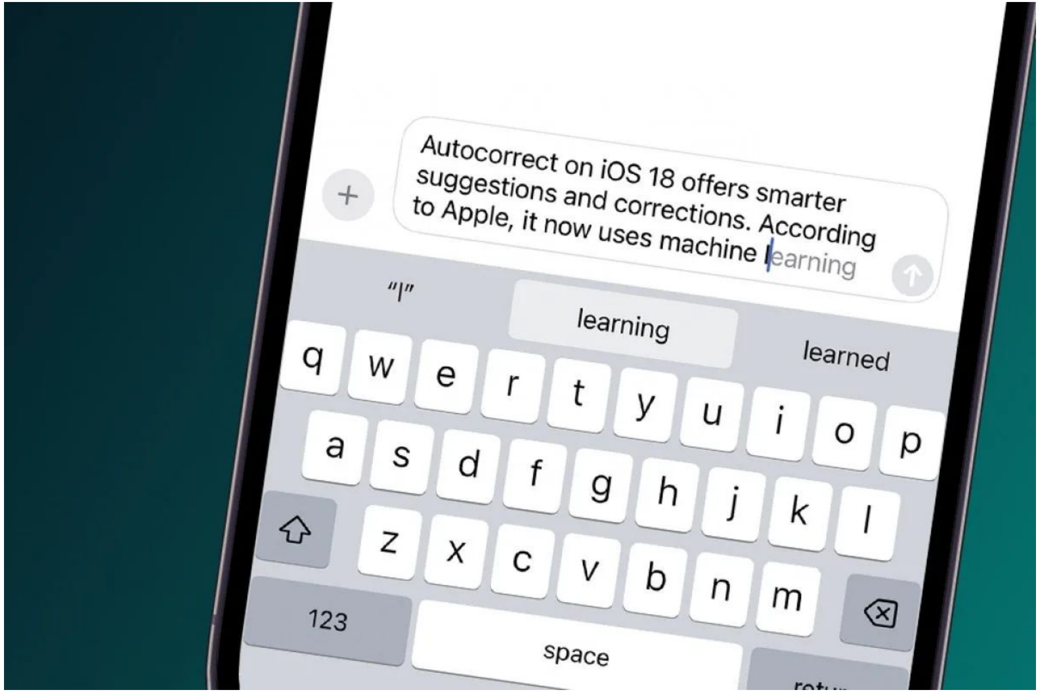
No need to explain timezone, invention of coffee..

With Artificial Intelligence



Example Use Cases of Language Models

- Autocomplete



Example Use Cases of Language Models

- Autocomplete

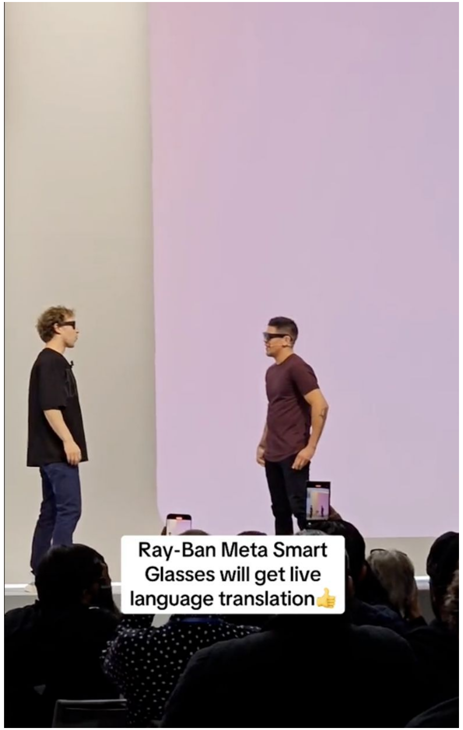
‘Let’s grab a…’ → Model guesses

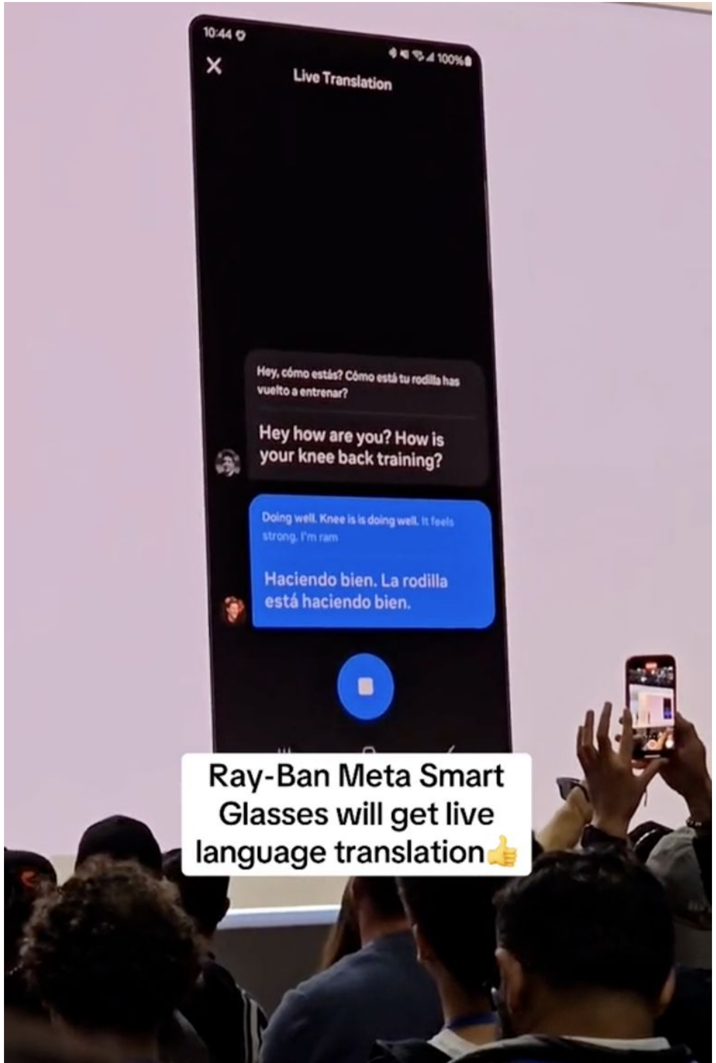
- beer?
- coffee?
- lunch?
- chainsaw?
- …

Next word prediction

Example Use Cases of Langu

- Machine Translation





Example Use Cases of Language Models

- Autocomplete

‘Bonjour, je m’appelle Marie.’ → Model guesses → ‘Hello, my name is Marie.’?

Actually not swapping word one by one. The model has to capture meaning in context.

Example Use Cases of Language Models

- Chatbots

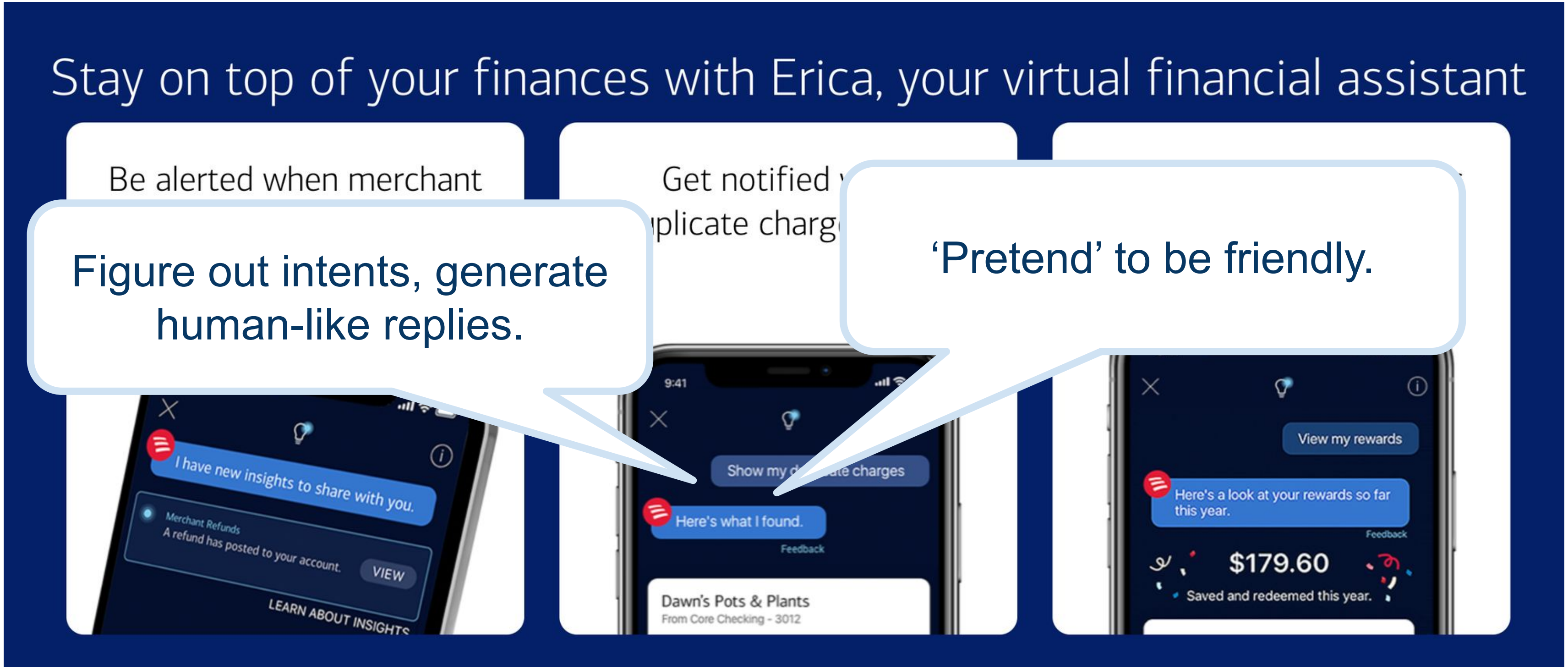
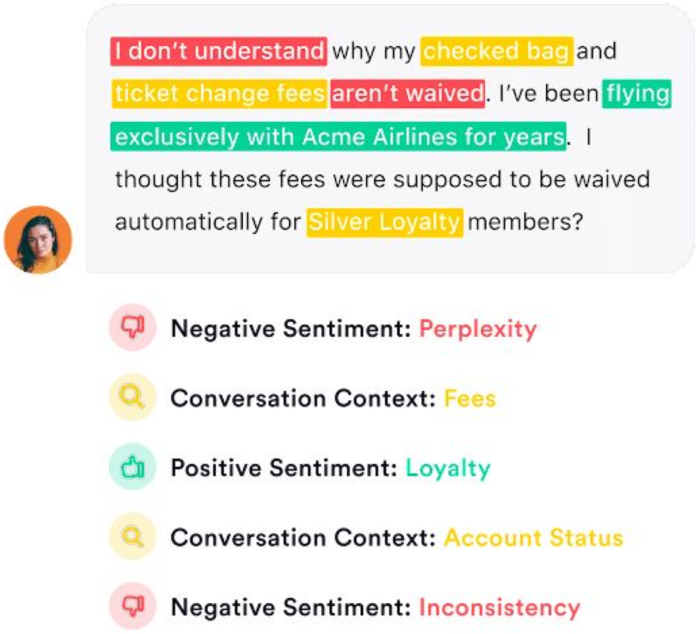


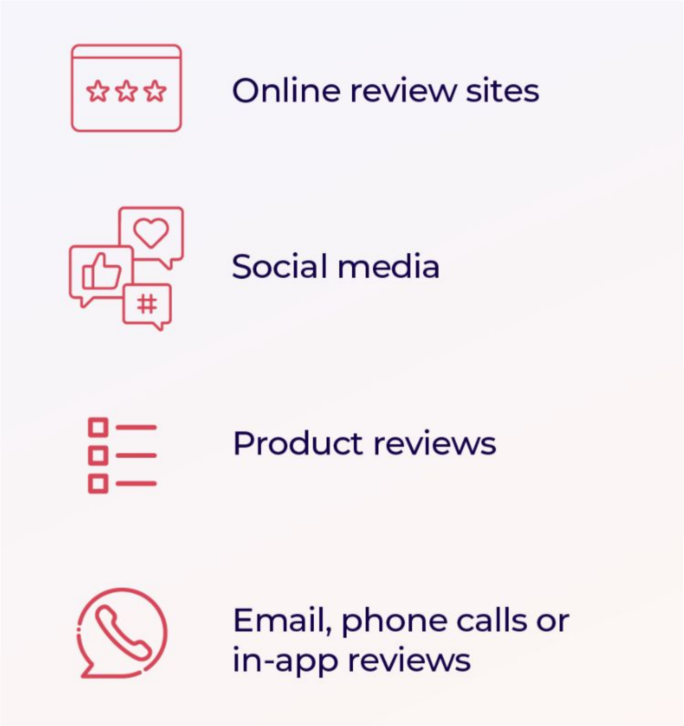
Figure out intents, generate human-like replies.

'Pretend' to be friendly.

Example Use Cases of Language Models

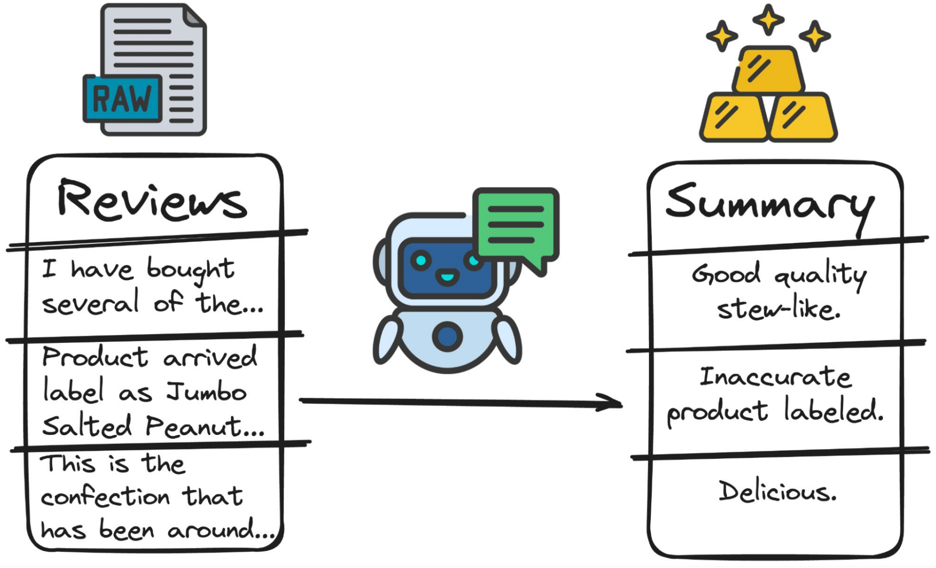
- Customer Feedback Analysis

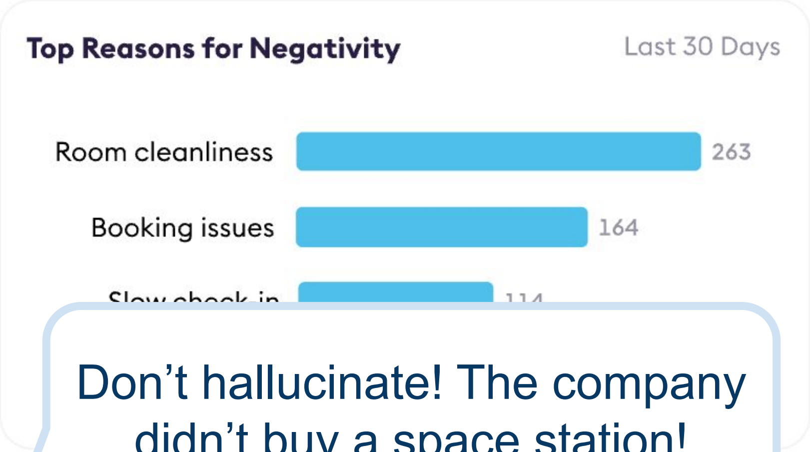




Example Use Cases of Language Models

- Automating reports



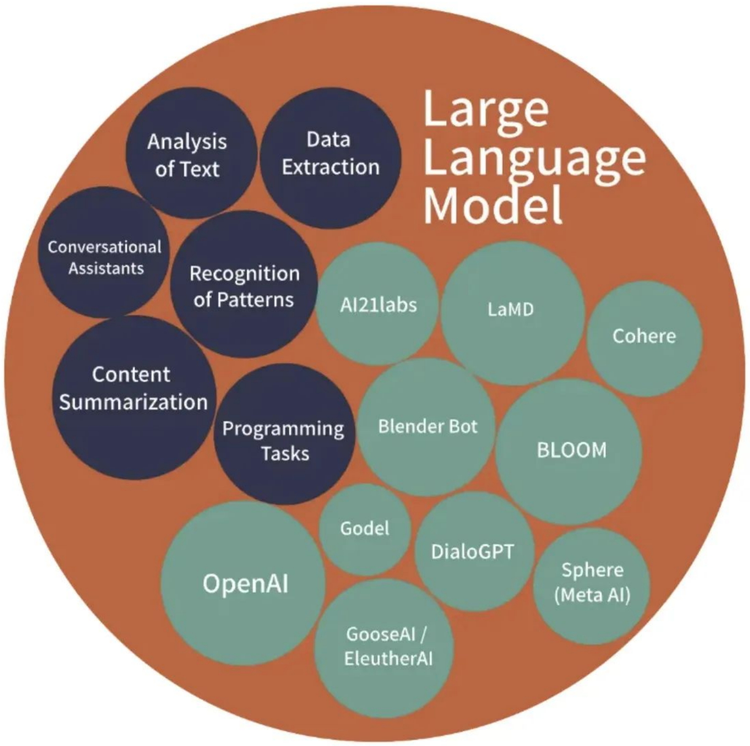


Don’t hallucinate! The company didn’t buy a space station!

‘Our [profits] increased by [percentage] thanks to [reason].’

Large Language Models

- GPT: Universal models - Has knowledge of the world?



Generating text: One word at a time.
Give it a few examples, prompt it.
Seem to understand the meanings of language.

S: I broke the window.
Q: What did I break?

S: I gave John flowers.
Q: Who did I give flowers to?

How many users have signed up since the start of 2020?
SELECT count(id)
FROM users WHERE created_at > ‘2020-01-01’

How do we represent the meaning of a word?

- Definition: meaning (Webster dictionary)
  - The idea that is represented by a word, phrase, etc.
  - The idea that a person wants to express by using words, signs, etc.
  - The idea that is expressed in a work of writing, art, etc

Symbol ⇔ Idea of something

tree ⇔ {🌳, 🌲, 🌴, ...}

How do we have usable meaning in a computer?

- Previously common NLP solution: Use, e.g., WordNet, containing lists of synonym sets and hypernyms (relationships)

e.g., synonym sets containing "good":

```
from nltk.corpus import wordnet as wn
poses = { 'n':'noun', 'v':'verb', 's':'adj (s)', 'a':'adj', 'r':'adv'}
for synset in wn.synsets("good"):
    print("{}: {}".format(poses[synset.pos()],
              ", ".join([l.name() for l in synset.lemmas()])))
```

```
noun: good
noun: good, goodness
noun: good, goodness
noun: commodity, trade_good, good
adj: good
adj (sat): full, good
adj: good
adj (sat): estimable, good, honorable, respectable
adj (sat): beneficial, good
adj (sat): good
adj (sat): good, just, upright
…
adverb: well, good
adverb: thoroughly, soundly, good
```

e.g., hypernyms of "panda":

```
from nltk.corpus import wordnet as wn
panda = wn.synset("panda.n.01")
hyper = lambda s: s.hypernyms()
list(panda.closure(hyper))
```

```
[Synset('procyonid.n.01'),
Synset('carnivore.n.01'),
Synset('placental.n.01'),
Synset('mammal.n.01'),
Synset('vertebrate.n.01'),
Synset('chordate.n.01'),
Synset('animal.n.01'),
Synset('organism.n.01'),
Synset('living_thing.n.01'),
Synset('whole.n.02'),
Synset('object.n.01'),
Synset('physical_entity.n.01'),
Synset('entity.n.01')]
```

How do we have usable meaning in a computer?

- A useful resource but missing nuance:
  e.g., “proficient” is listed as a synonym for “good”. Only correct in some contexts

Cases where “proficient” ≈ “good”

✅ She is a proficient piano player.

✅ He is proficient in Spanish.

Cases where “proficient” ≠ “good”

❌ He is a proficient person.

❌ That cake tastes proficient.

How do we have usable meaning in a computer?

- A useful resource but missing nuance:
  e.g., “proficient” is listed as a synonym for “good”. Only correct in some contexts

- Missing new meanings of words:
  e.g., wicked, bet, lit, fire

- Impossible to keep up to date with human labor.

How do we have usable meaning in a computer?

- **Traditional NLP**, we regard words as discrete symbols:
- One-hot coding:

motel, hotel, cat, on, the, mat, oh

These symbols can be represented by **one-hot vectors**:

motel = [1 0 0 0 0 0 0]
hotel = [0 1 0 0 0 0 0]
cat = [0 0 1 0 0 0 0]
on = [0 0 0 1 0 0 0]
the = [0 0 0 0 1 0 0]
mat = [0 0 0 0 0 1 0]
oh = [0 0 0 0 0 0 1]

Vectors constituting only of 1 and 0

Each word in vocabulary is assigned a unique index.

Each word is represented as a vector of all zeros, expect for a 1 in its position.

Dimension is the length of the vocabulary.

How do we have usable meaning in a computer?

- **Traditional NLP**, we regard words as discrete symbols:
- One-hot coding:

'Orthogonal', not relevant or similar at all.

motel	=	[1 0 0 0 0 0 0]
hotel	=	[0 1 0 0 0 0 0]
cat	=	[0 0 1 0 0 0 0]
on	=	[0 0 0 1 0 0 0]
the	=	[0 0 0 0 1 0 0]
mat	=	[0 0 0 0 0 1 0]
oh	=	[0 0 0 0 0 0 1]

E.g. If I'm searching for 'cat hotel', we would also like to find documents containing 'cat motel'

No natural notion of 'similarity'

Representing words by their context

- **Modern DL Idea:** A word's meaning is given by the words that frequently appear close-by.

"You shall know a word by the company it keeps" (J. R. Firth 1957: 11)

Representing words by their context

- When a word appears in a text, its **context** is the set of words that appear nearby.

...government debt problems turning into ***banking*** *crises as happened in 2009...*
...saying that Europe needs unified ***banking*** *regulation to replace the hodgepodge...*
...India has just given its ***banking*** *system a shot in the arm...*

These context words will represent ***banking***

Representing words by their context

- A word's meaning is reflected by the words that tend to appear near it in text.

"The **cat** chased the mouse."

"The **dog** barked at the cat."

"A **lion** is chasing another animal."

Words like cat, dog, lion often share context words (animal, chase, fur, bark, etc.).

In raw data, their "neighbors" overlap.

Word Embeddings

- We want to adjust word vectors so that:
  - If two words show up in **similar contexts**, their vectors move **closer** together.
  - If they rarely appear in the same context, their vectors move further apart.

$$\textit{banking} = \begin{pmatrix} 0.286 \\ 0.792 \\ -0.177 \\ -0.107 \\ 0.109 \\ -0.542 \\ 0.349 \\ 0.271 \end{pmatrix} \qquad \textit{monetary} = \begin{pmatrix} 0.413 \\ 0.582 \\ -0.007 \\ 0.247 \\ 0.216 \\ -0.718 \\ 0.147 \\ 0.051 \end{pmatrix}$$

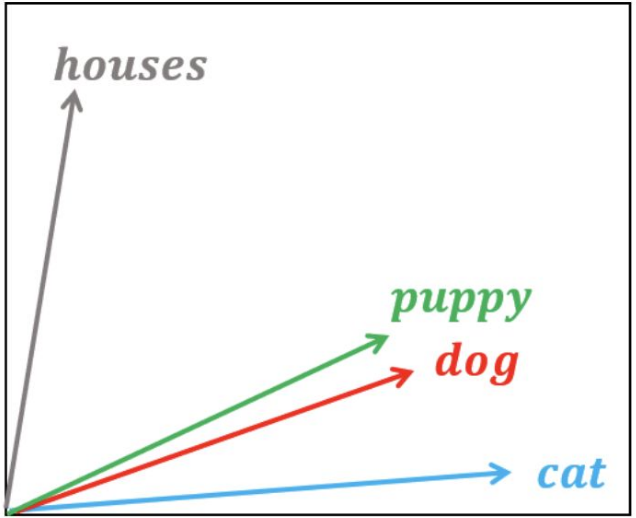
Dense word vectors / **Embeddings**:
- Not one-hot anymore!
- Each number encodes some aspect of meaning

‘Closer’

- when we project embeddings into 2D space (for visualization), we see neat clusters:

	d1	d2	d3	d4	d5	d6	d7
dog →	0.6	0.9	0.1	0.4	−0.7	−0.3	−0.2
puppy →	0.5	0.8	−0.1	0.2	−0.6	−0.5	−0.1
cat →	0.7	−0.1	0.4	0.3	−0.4	−0.1	−0.3
houses →	−0.8	−0.4	−0.5	0.1	−0.9	0.3	0.8

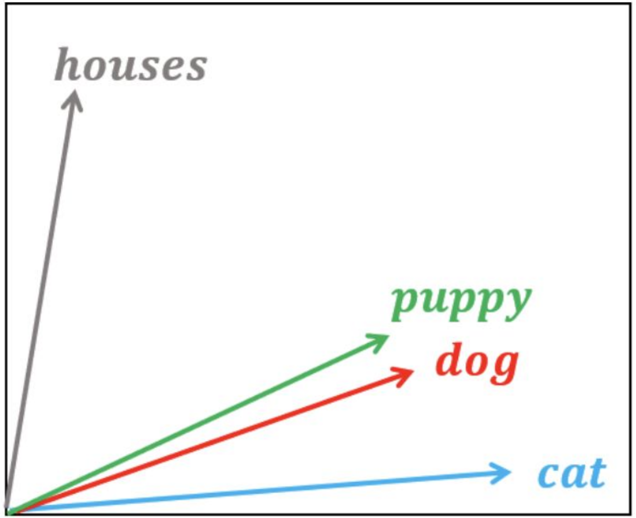
Dimensionality reduction of word embeddings from 7D to 2D →



The vectors of puppy and dog are **‘closer’**

‘Closer’

- when we project embeddings into 2D space (for visualization), we see neat clusters:



$$\text{cosine similarity}(u, v) = \frac{u \cdot v}{\|u\|\|v\|} = \cos(\theta)$$

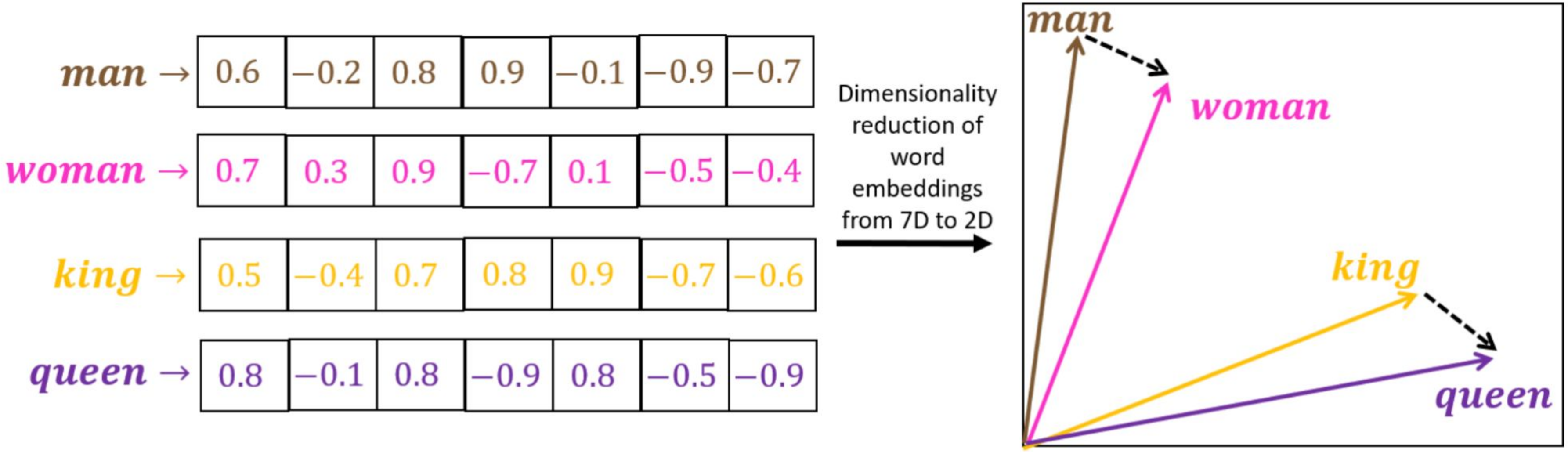
“How much do these two arrows point in the same direction?”

Q: Let’s think about one-hot coding, cosine similarity between vectors?

(θ = 90°), cos(θ) = 0 → similarity = 0 (no similarity).

‘Closer’

- The embeddings should also captures relational meanings.



Visualize Dense Embeddings

- https://projector.tensorflow.org/

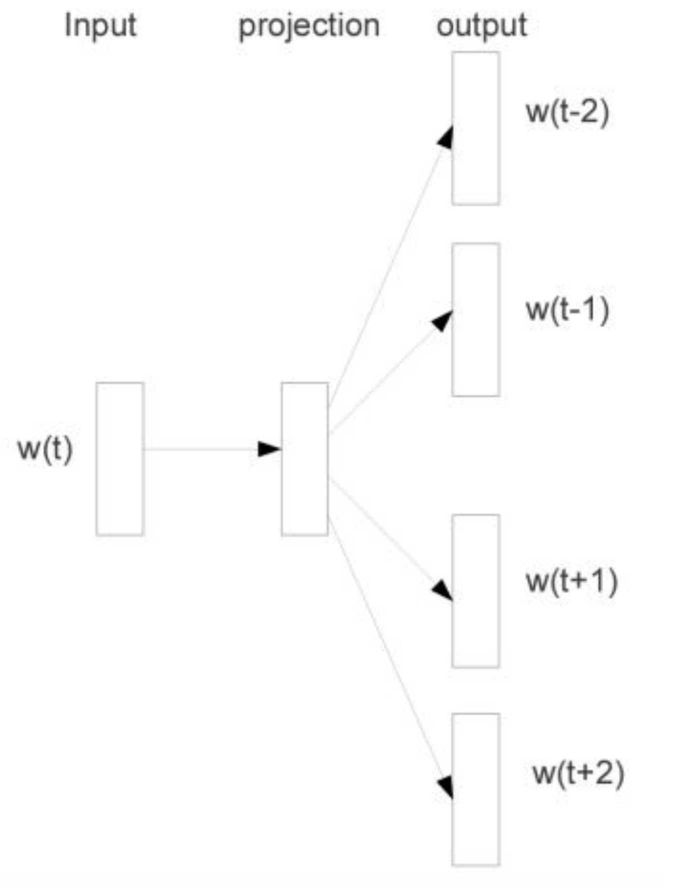
Word2Vec

- Word2Vec is a framework that learns embeddings by looking at the context in which words appear (Mikolov et al., 2013) .

Idea:

- We have a large corpus (“body”) of text: a long list words.
- Every word in a fixed vocabulary is represented by vector.
- **Predicts the context words from a target word.**

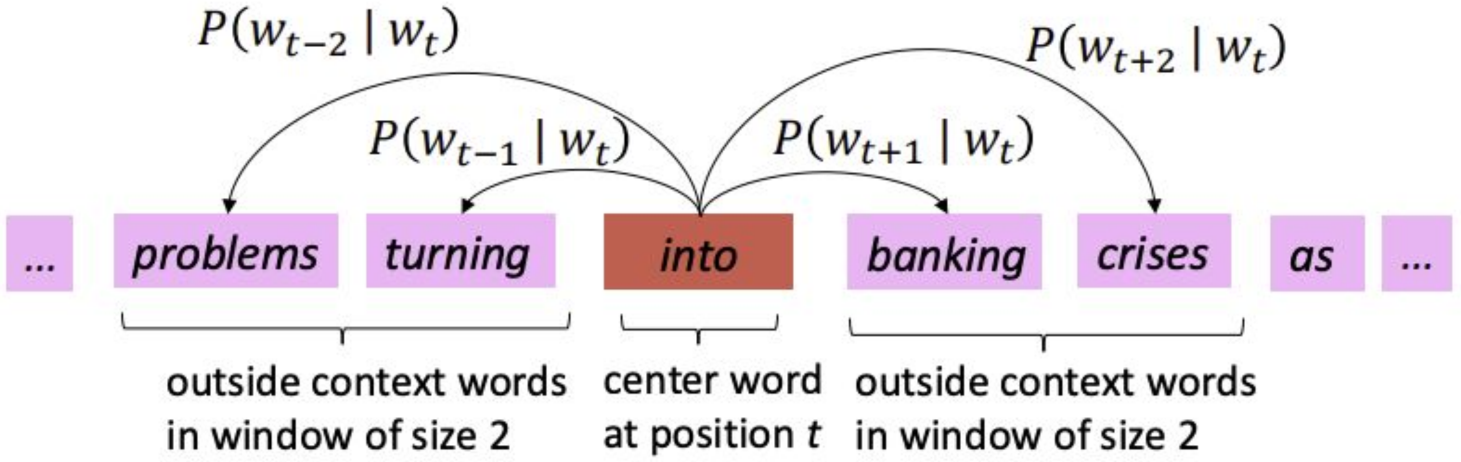
  Example: Input = “cat.”
  Predict: “the,” “chased,” “mouse.”



Skip-gram model
(Mikolov et al. 2013)

Word2Vec

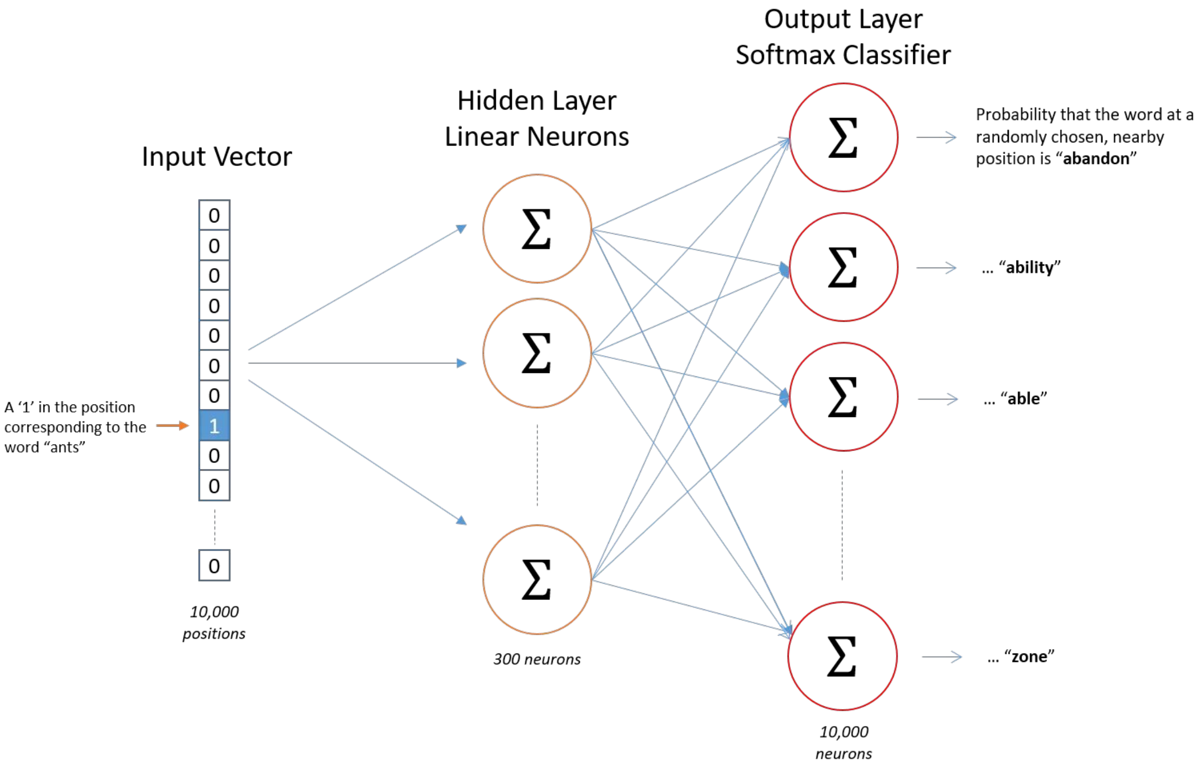
Idea:

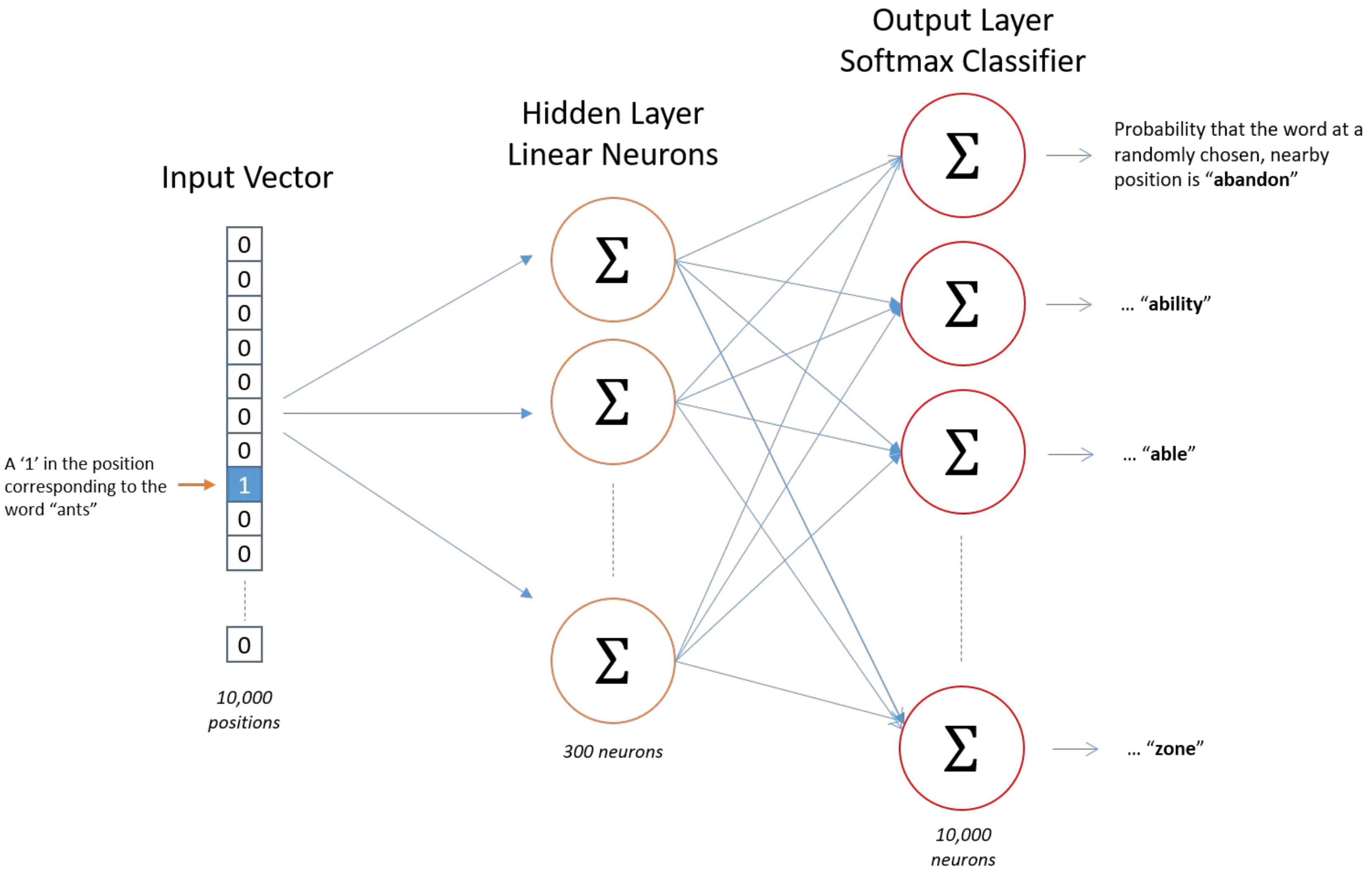


Word2Vec Architecture

- A NN with one hidden layer.
- Inputs: All the documents/texts in our training set, represented in 1-hot encoding.

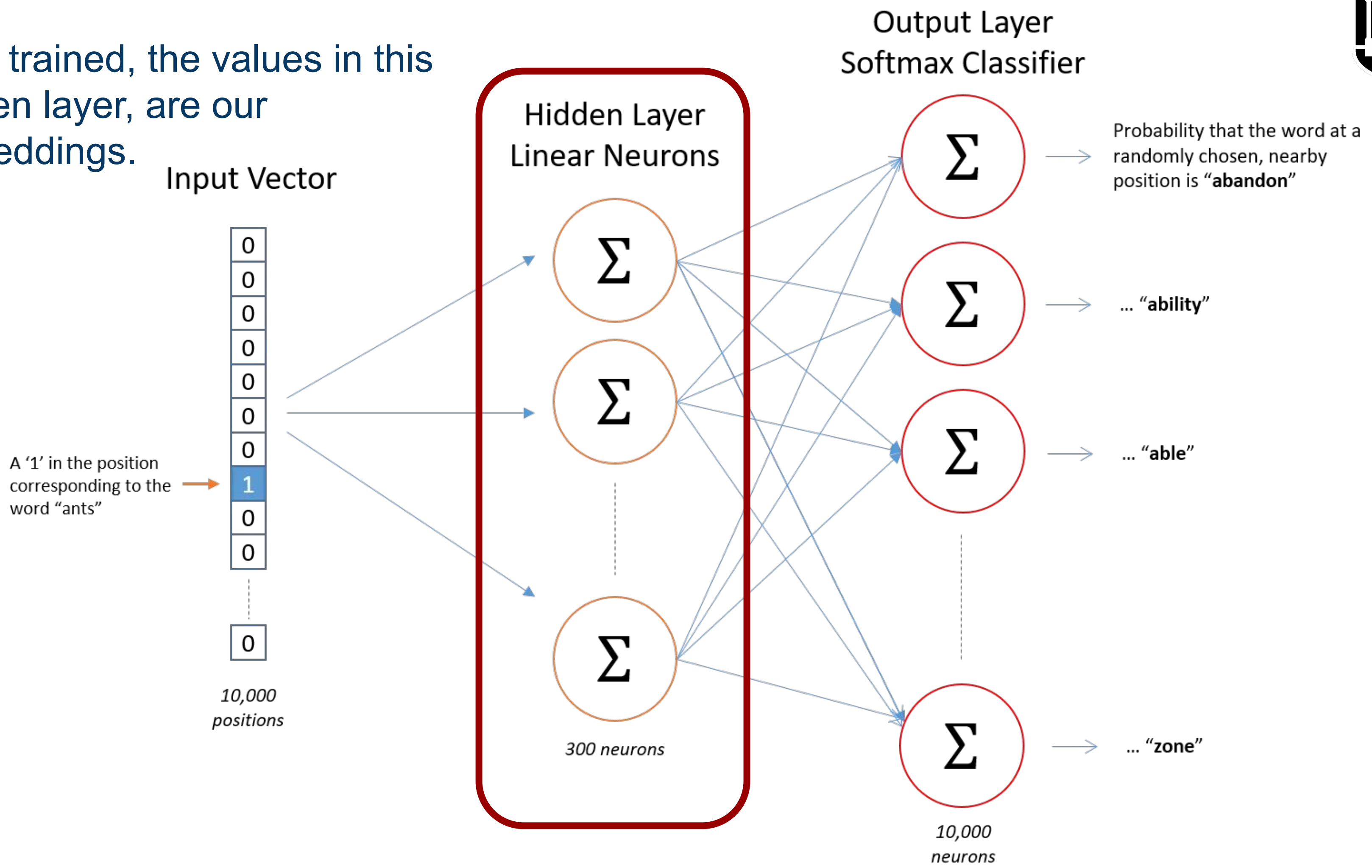
One hidden layer, dimension: equal to the length of the embedding we want





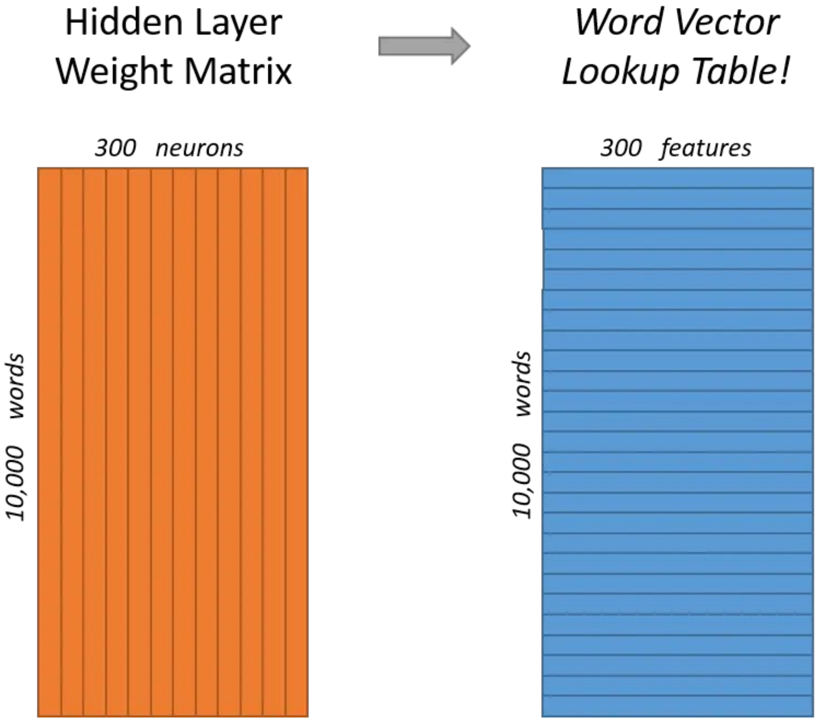
Output Layer
Softmax Classifier
Hidden Layer
Linear Neurons
Input Vector
0
0
0
0
0
0
0
1
0
0
0
A '1' in the position corresponding to the word "ants"
10,000 positions
Σ
Σ
Σ
300 neurons
Σ
Σ
Σ
Σ
10,000 neurons
Probability that the word at a randomly chosen, nearby position is "abandon"
… "ability"
… "able"
… "zone"

- After trained, the values in this hidden layer, are our embeddings.



Word2Vec

- After training, passing through a one-hot coded word, the first hidden layer vector is our embedding!



Word2Vec Training

- We want to maximize the probability of seeing the context word ooo given the center word.
- Equivalent to maximizes objective function by putting similar words nearby in space

Another Architecture