11 – Natural Language Processing

André Lamúrias

Natural Language Processing

Natural Language Processing

- Use of human languages by a computer
- Different from computer languages – ambiguous, variability, inconsistency, tone, etc
- Applications in machine translation, chatbots, information retrieval
- Language models – probability distribution over sequences of words

Natural Language Processing

- Can use both symbolic and sub-symbolic AI
- Machine Learning can be used for NLP
- Challenges:
    - Sequential data
    - High dimensional data – many words
- One-hot encoding leads to very sparse data
    - Most words are not used – vectors are mostly 0s

NLP concepts

- Tokens – smallest unit used in NLP
  - Words, characters, or parts of words (subwords)
- Token -> Sentence -> Document –> Corpus
- Lemma/Stem – root of the word
  - Remove suffixes and conjugation e.g. is->be, involves-> involve
- Part-of-speech: Class of the word
  - E.g. noun, verb, adjective
- Tokenization – process of splitting text into token units
- Sentence splitting
- Stop words – very commonly used words
  - The, that, is, a, …

N-gram models

N-grams

- Given h=*"its water is so transparent that"*
  - How to calculate P("the"|h)?
  - Take a large corpus, count the number of times we see h and how often it is followed by "the"

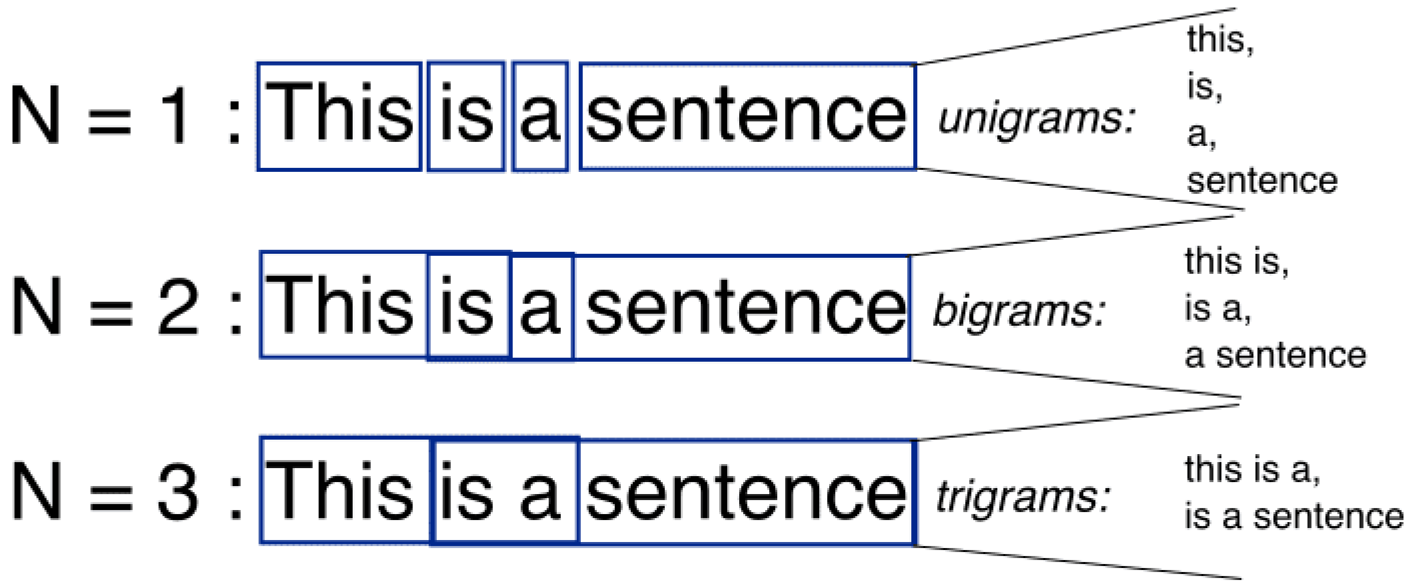
$$P(\texttt{the}|\texttt{its water is so transparent that} = \frac{C(\texttt{its water is so transparent that the})}{C(\texttt{its water is so transparent that})}$$

  - What if it is a new sentence?

N-grams

- N-gram – sequence of N tokens
- N-gram models: predict next token given a sequence of tokens
- N=1 -> Unigrams/Bag-of-words: each token has a fixed probability
- N=2 -> Bi-gram model
  - Given one word, predict the next one
  - We can count how many times each token occurs after another token
- N=3 -> Tri-gram model
  - Given two consecutive words, predict the next one
  - We can count how many times each token occurs after those two words

N-grams



Source: https://www.kdnuggets.com/2022/06/ngram-language-modeling-natural-language-processing.html

N-grams

- How to calculate joint probability of a sequence of words?
  - Use chain rule of probability

$$
\begin{aligned}
P(X_1 ... X_n) &= P(X_1)P(X_2|X_1)P(X_3|X_{1:2})...P(X_n|X_{1:n-1}) \\
&= \prod_{k=1}^{n} P(X_k|X_{1:k-1})
\end{aligned}
$$

- We need the conditional probability of a token given its previous tokens
- We approximate by using only n-1 previous words instead of all previous words

Bigram models

- We approximate P(the | water is so transparent that) with P(the|that)

$$P(w_n|w_{1:n-1}) \approx P(w_n|w_{n-1})$$

- We ca generalize to other n-grams (N is the n-gram size)

$$P(w_n|w_{1:n-1}) \approx P(w_n|w_{n-N+1:n-1})$$

Bi-gram models

- Now we can compute the probability of a word sequence

$$P(w_{1:n}) \approx \prod_{k=1}^{n} P(w_k | w_{k-1})$$

- To get these probabilities we count and normalize so that the sum is 1
- We augment sentences with a special start and end sentence symbols

Example

<s> I am Sam </s>
<s> Sam I am </s>
<s> I do not like green eggs and ham </s>

$P(\texttt{I} \mid \texttt{<s>}) =$ [redacted] $P(\texttt{Sam} \mid \texttt{<s>}) =$ [redacted] $P(\texttt{am} \mid \texttt{I}) =$ [redacted]

$P(\texttt{</s>} \mid \texttt{Sam}) =$ [redacted] $P(\texttt{Sam} \mid \texttt{am}) =$ [redacted] $P(\texttt{do} \mid \texttt{I}) =$ [redacted]

N-gram models

- Issues:
  - Longer n-grams – bigger matrices
  - Unseen n-grams: count is zero
    - What if it appears on the test set?
    - Model smoothing – add fake count
  - Unknown words (out-of-vocabulary – UNK token)
  - Unidirectional, not very generalizable

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.

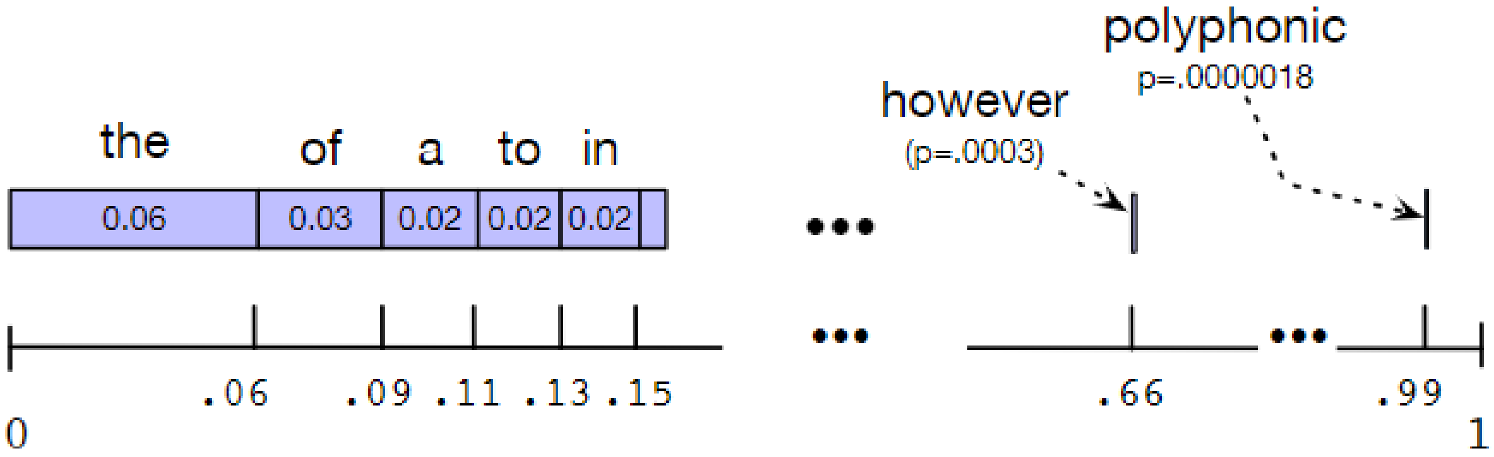
Evaluating LMs

- Extrinsic evaluation – next lecture
- Intrinsic evaluation: Perplexity (PP or PPL)
  - According to the model, how surprising is a sequence of tokens?
  - Inverse probability divided by number of words
  - May not correlate with improvement in the task

$$
\begin{aligned}
\text{perplexity}(W) &= P(w_1 w_2 \ldots w_N)^{-\frac{1}{N}} \\
&= \sqrt[N]{\frac{1}{P(w_1 w_2 \ldots w_N)}}
\end{aligned}
$$

Generating text

- Sampling from a LM
  - We generate sentences that have high probability according to the model
  - We sample tokens according to their probability, given its previous n-1 words
    - Ends when end of sentence token is sampled



Neural Language Models

Neural Language Models

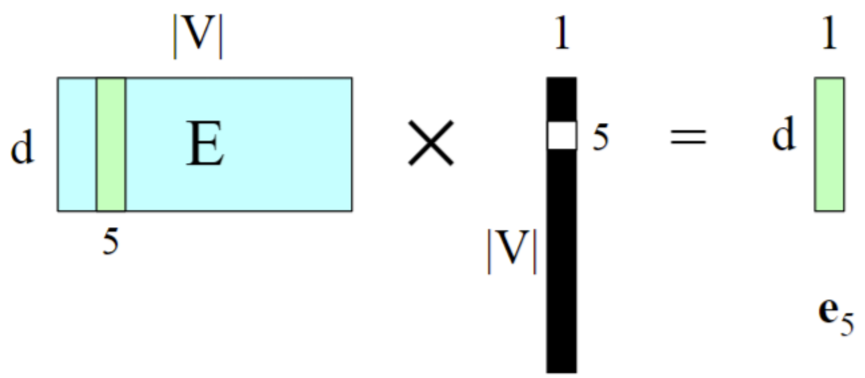
- Predict next word – now using Neural Networks instead of n-gram probabilities
- Token are represented by embeddings
  - This way we can predict unseen combinations of tokens
- First we represent words with One-hot vectors

```
[0 0 0 0 1 0 0 ... 0 0 0 0]
 1 2 3 4 5 6 7 ...   ... |V|
```

- Where V is the vocabulary, and this word is the 5th in the vocabulary

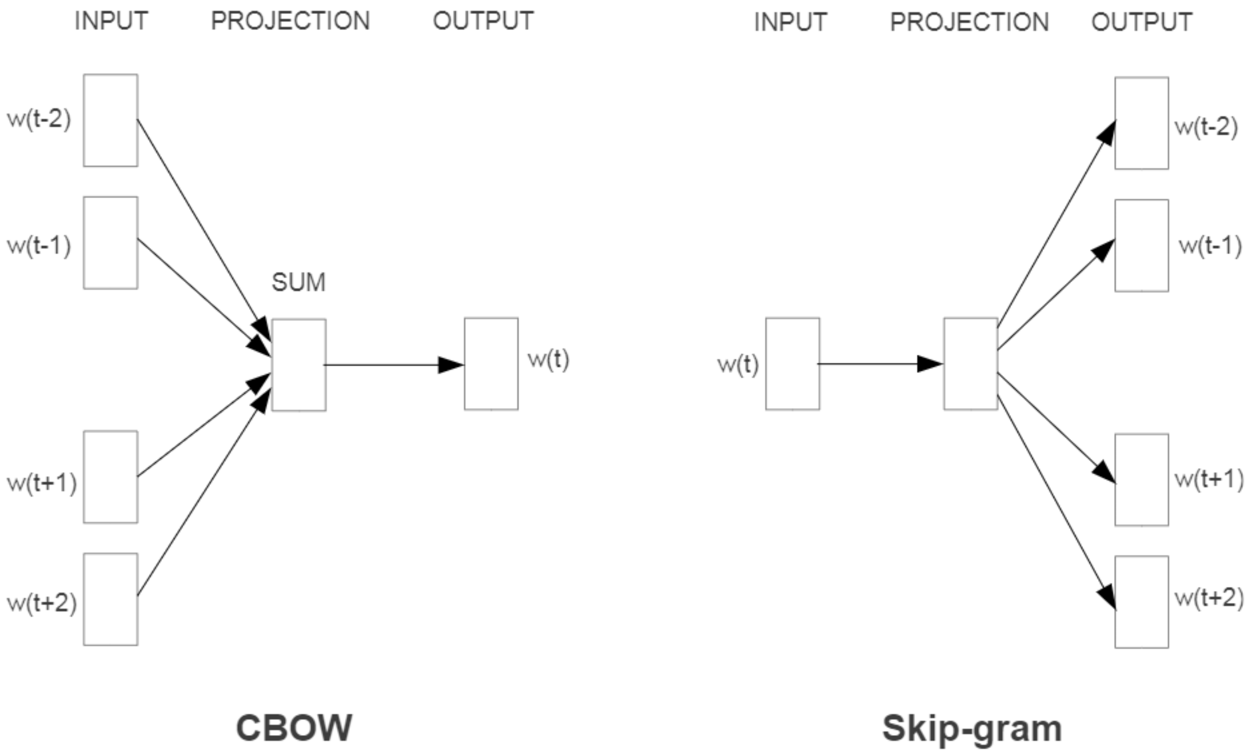
Embeddings

- Embedding matrix – features of each token of the vocabulary
  - Each column is a token, in order
  - Number of lines d is a hyperparameter
  - Dense representation of words



Embeddings – Word2Vec

- Distinguish between words that are in the context of another words
    - Positive examples from dataset
    - Negative examples randomly sampled
- Logistic regression
- Static embeddings



Skip-Gram Training data

...lemon, a [tablespoon of apricot jam, a] pinch...

- c1 c2 [target] c3 c4

positive examples +

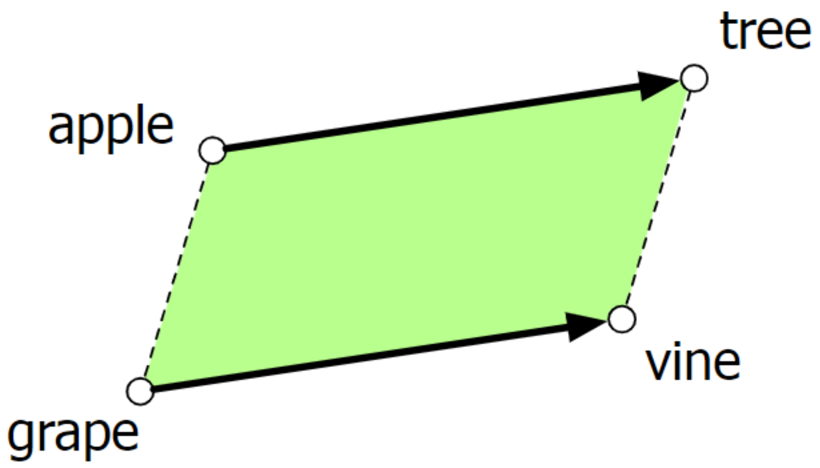
t	c
apricot	tablespoon
apricot	of
apricot	jam
apricot	a

negative examples -

t	c	t	c
apricot	aardvark	apricot	seven
apricot	my	apricot	forever
apricot	where	apricot	dear
apricot	coaxial	apricot	if

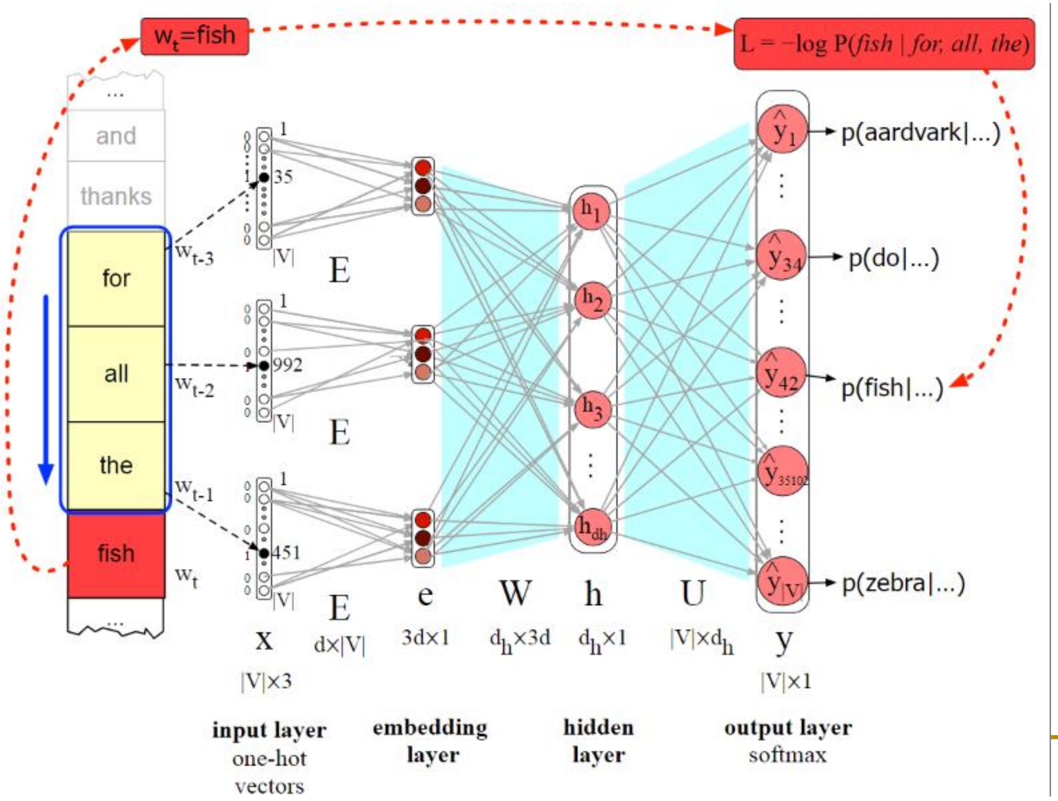
Embedddings – Relational Similarity

- king – man + woman = queen
- Paris – France + Italy = Rome
- https://code.google.com/archive/p/word2vec



Next word prediction

- Use softmax to obtain probability of all words in the vocabulary, given the input words



Training LMs

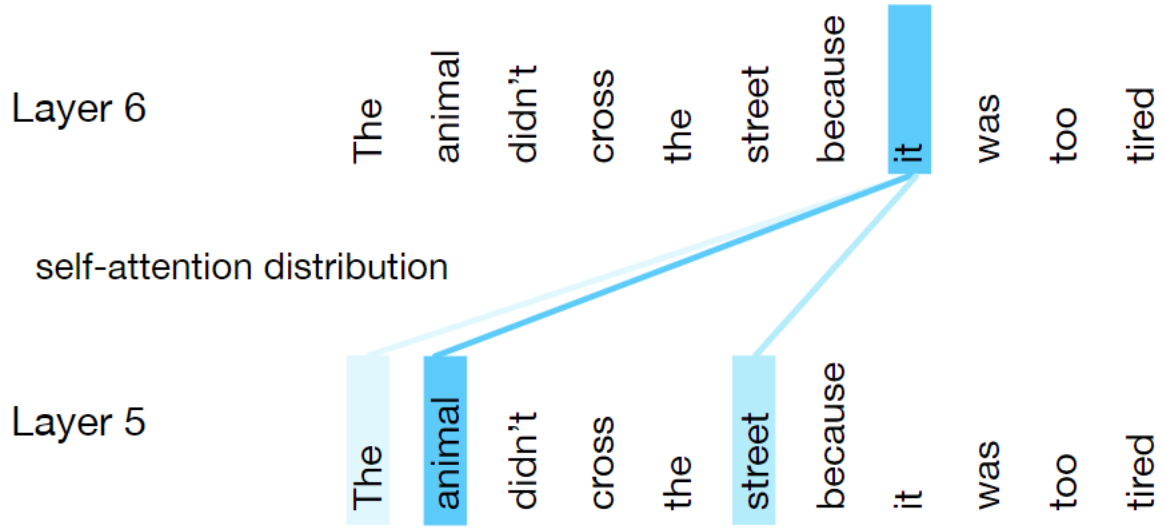
- Self-supervision using a corpus of text
  - We always know the next word in the training data
  - Maximize the probability of that next word being the right one
  - Same as minimizing negative log likelihood
- Backpropagate all the way to the embedding layer
  - Randomly initialized

Transformers and Large Language Models

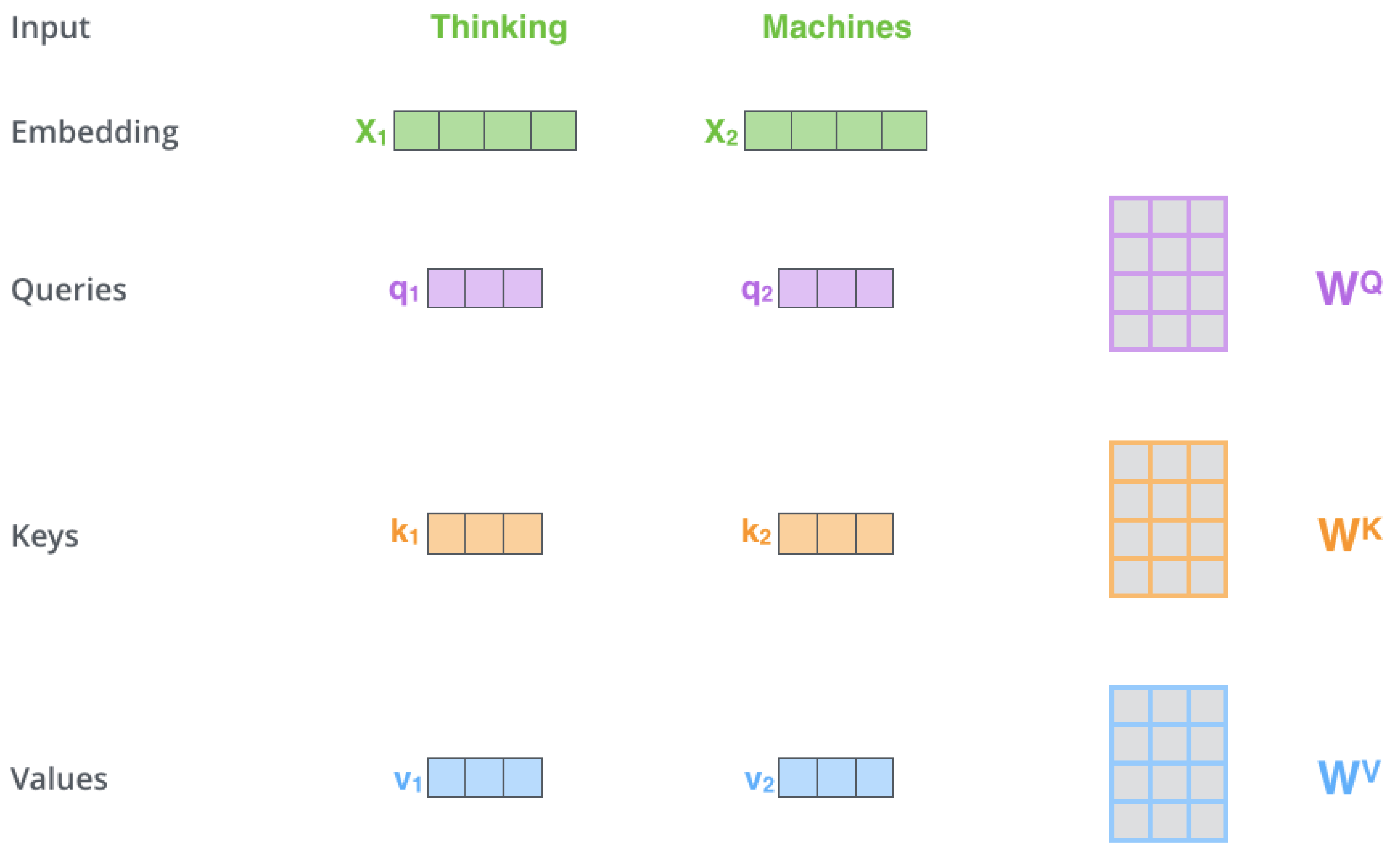
- Idea – instead of pre-training embedding layer, pre-train full NN for contextual embeddings
- What architecture should this model have?
  - Need to handle long distance relations
  - But needs to be more efficient than recurrent networks
- Transformers' main innovation – self attention layers

Self-attention

- At each layer, produce contextual representation of the words
  - Therefore, we need to take into account the neighbors of each word

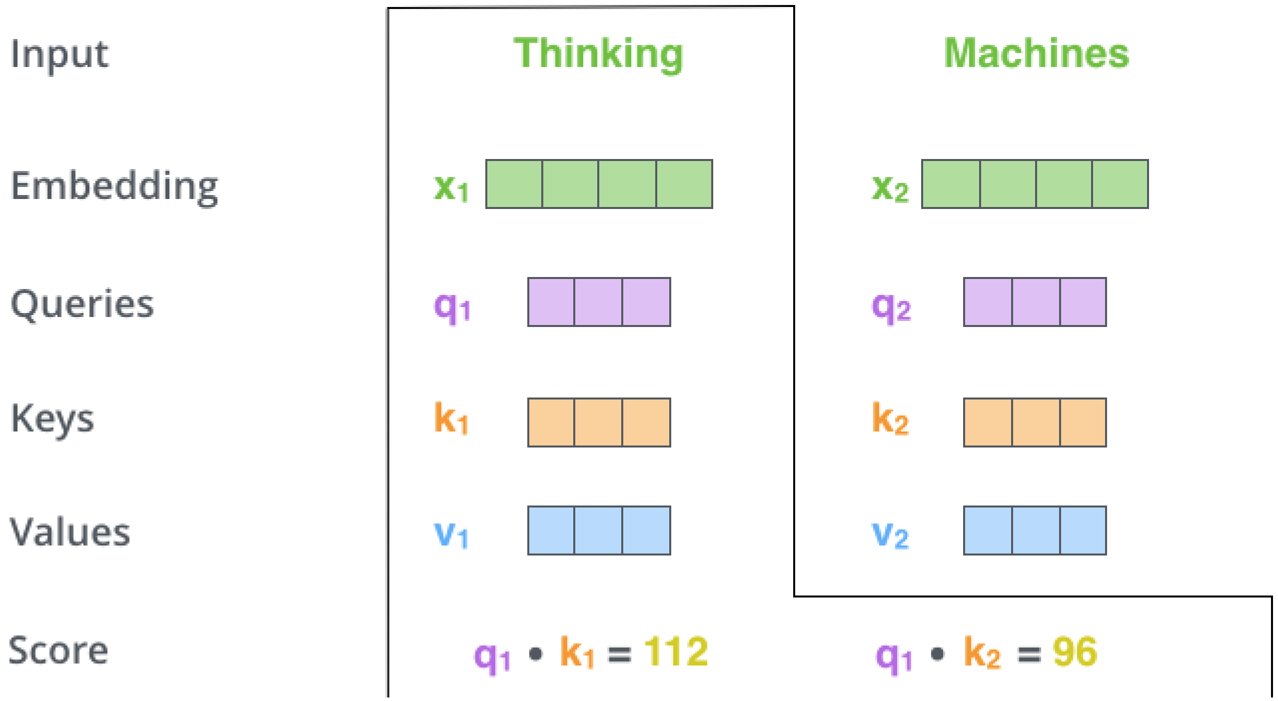


Self-attention mechanism



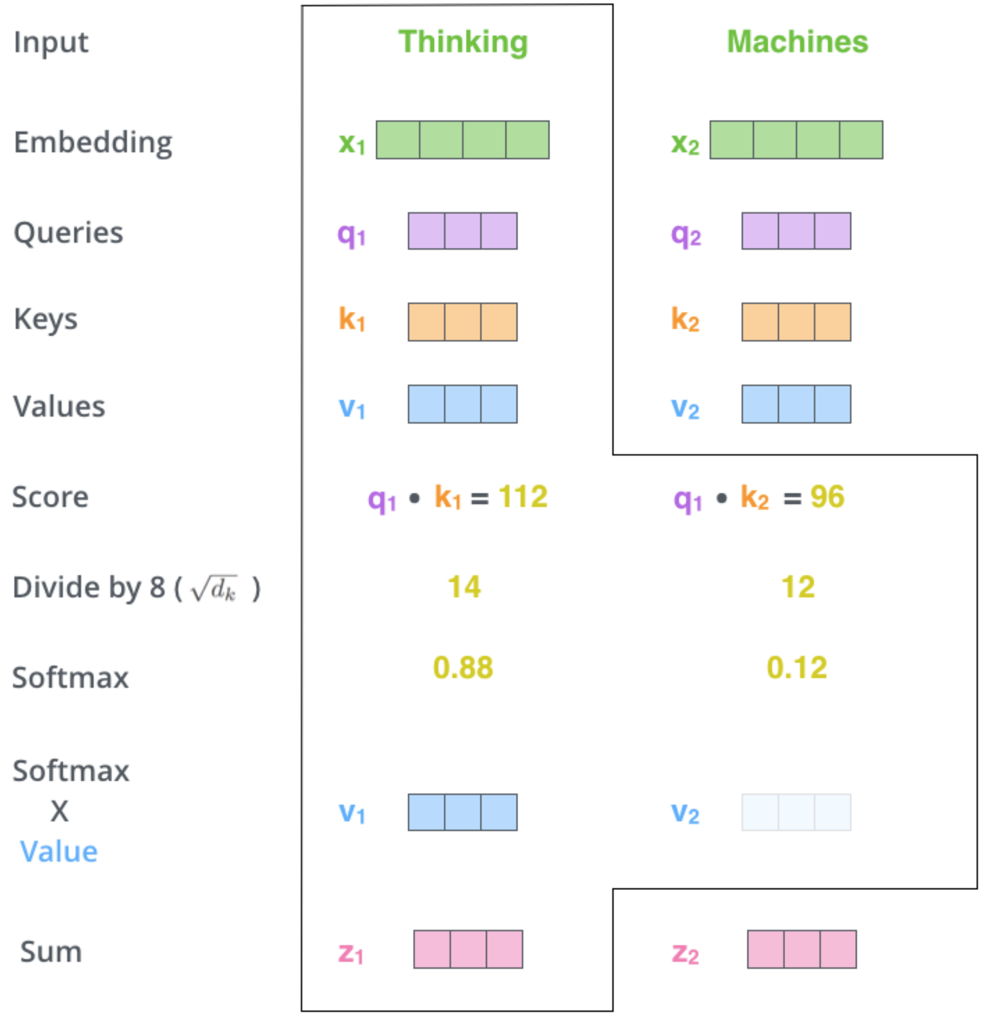
Source: https://jalammar.github.io/illustrated-transformer/

Self-attention mechanism



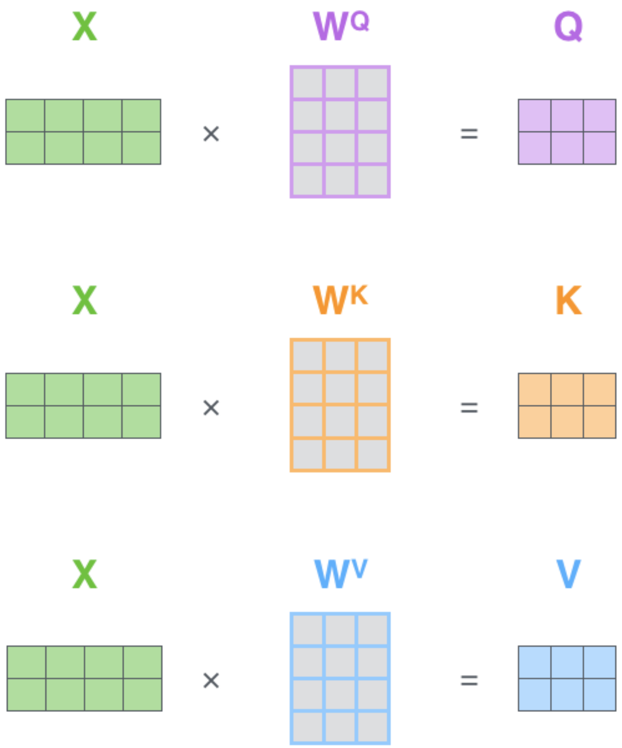
Source: https://jalammar.github.io/illustrated-transformer/

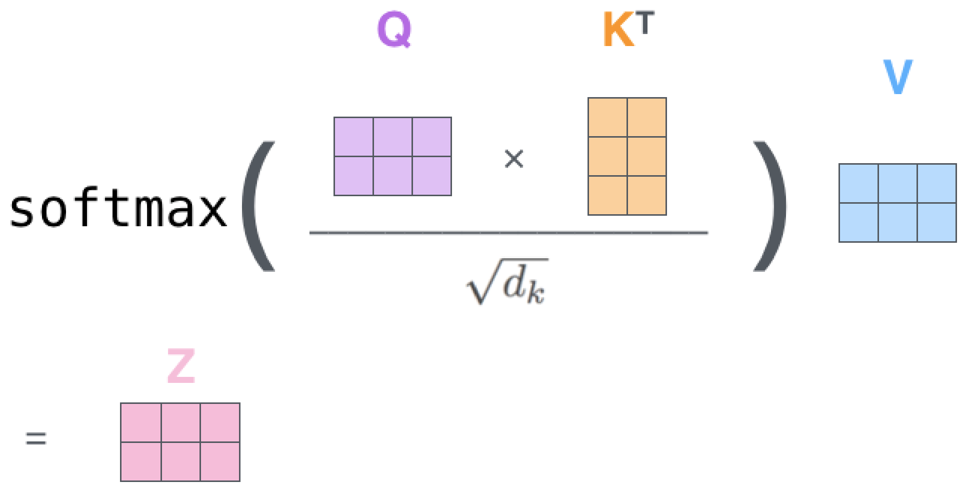
Self-attention mechanism



Self-attention mechanism

- We can do this quickly with matrix multiplication



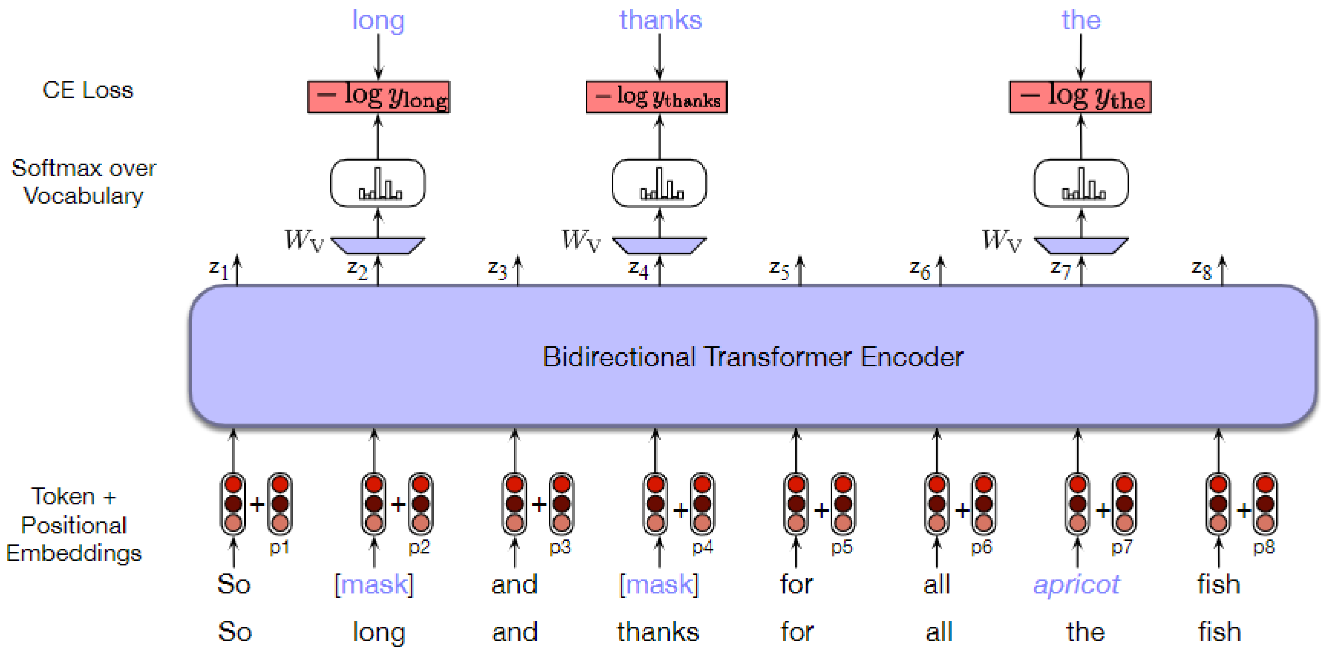


Improvements

- Multi-head attention: multiple Q, K and V matrices
  - Each head can learn different relations between words
- Order is represented with positional embeddings
  - Otherwise, the transformer model does not care about word order
- Explore attention: https://huggingface.co/spaces/exbert-project/exbert

Training Transformers

- Masked Language Modeling:
  - Randomly pick tokens to replace with special [MASK] token (or random word)
  - Do this for 15% of the tokens
  - Predict original token
- Next sentence prediction
  - Predict if sentences are related or not



Prompting and LLMs

- Many NLP tasks can be done with next word prediction
- E.g. “The sentiment of the sentence “I like Jackie Chan” is”
  - Compare prob of positive and negative
- E.g. “Q: Who wrote the book “The Origin of Species"? A:”
  - Look most likely next words
  - Could be wrong!
- Current LLMs (like ChatGPT) have additional layers to improve their answers

Summary

- Natural Language Processing
- N-gram models
- Neural linguistic models
- Further reading:
  - Goodfellow, chapter 12.4
  - “The spelled-out intro to language modeling: building makemore”
  - https://www.youtube.com/watch?v=PaCmpygFfXo
  - Speech and Language Processing Chapters 3, 7 and 10
  - https://jalammar.github.io/illustrated-transformer/