Large Language Models

Where do they come from ? an historical perspective

Benoît Crabbé

INED, January 12th 2024

Université Paris Cité

Outline

1. Early Machine Translation
2. Symbolic NLP
3. Language models
4. Deep learning language models

The Cold War

- Alan Turing (”Computing Machinery and Intelligence”, *Mind*, 1950) is the first to ask *Can machines think ?*, W. Weaver (1949) *Translation* and C. E. Shannon published ”Prediction and entropy of Printed English”, 1951
- The first projects in natural language processing (NLP) started during the cold war.
  - The goal was to translate Russian into English with computers.
  - The method relies on exhaustive dictionaries (word to word translation)
- The problem that showed up immediately is **ambiguity**

Example (Machine translation)

The spirit is willing but the flesh is weak

... translate to Russian then back into English ...

The vodka is strong but the meat is rotten

Translation

(W. Weaver 1949)

- Machine translation can be automated
- Ambiguity can be solved thanks to context:

  If one examines the words in a book, one at a time through an opaque mask with a hole in it one word wide, then it is obviously impossible to determine, one at a time, the meaning of words. " Fast" may mean "rapid"; or it may mean "motionless"; and there is no way of telling which. But, if one lengthens the slit in the opaque mask, until one can see not only the central word in question but also say N words on either side, then, if N is large enough one can unambiguously decide the meaning…
- Cryptographic methods can be used for translation

  One naturally wonders if the problem of translation could conceivably be treated as a problem in cryptography. When I look at an article in Russian, I say: This is really written in English, but it has been coded in some strange symbols. I will now proceed to decode. (Weaver 1947)
- Meaning can be represented independently of any specific language

Solving ambiguity…is hard

Example (Machine translation)

How to translate *pen* in examples such as:

Little John was looking for his toy box. Finally he found it. The box was in the pen.

1. La boite était dans le stylo (deepL – 2023)
2. La boite était dans l'enclos (me)

The Bar Hillel (1960) and ALPAC (1966) Reports

- The first experiments in computational language modelling highlighted the importance of common sense and implicit knowledge of the world and knowledge of language
- Bar-Hillel (1960) "The present status of automatic translation of languages"
  - **Conclusion:** Fully automated machine translation is an illusion and the only reasonable objective is Human aided automatic translation

Outline

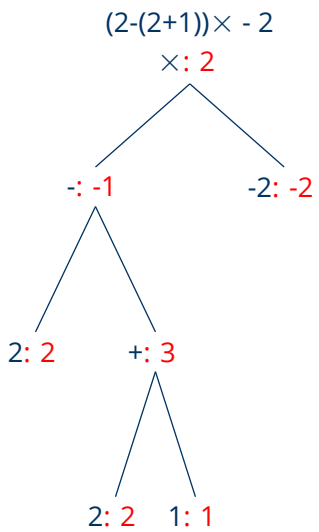
1. Early Machine Translation
2. Symbolic NLP
3. Language models
4. Deep learning language models

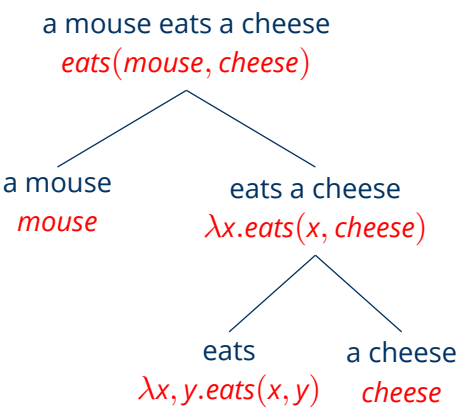
The rise of symbolic AI and generative linguistics (1956)

- **1956**: Dartmouth summer school (John Mc Carthy, Marvin Minsky, Allan Newell, Claude Shannon, Herbert Simon...)
- **1956**: "Three models for the description of language" (Chomsky). The ALPAC report also legitimates the generative endeavor and computational linguistics as the new linguistics automating generative grammar

Symbolic AI

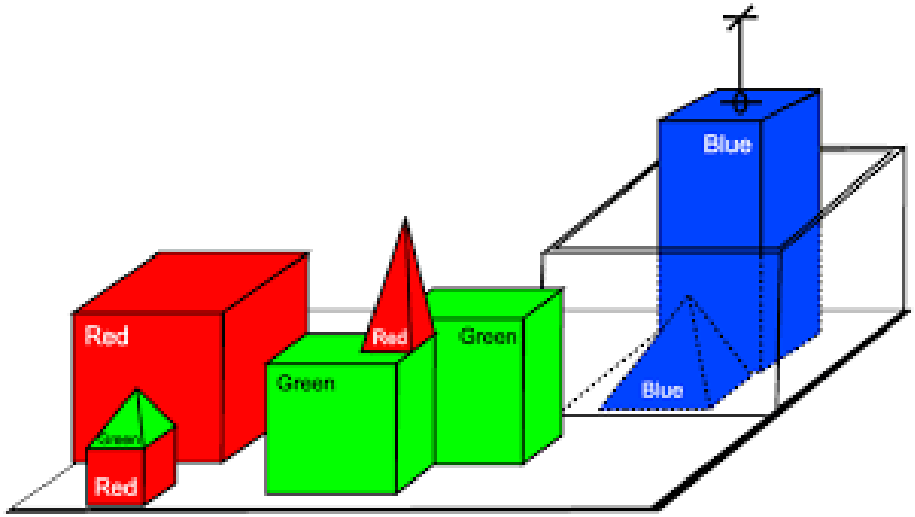
That's the golden age of symbolic AI that begins. In NLP, beyond Chomsky's work on formalizing language **structure** and works from R. Montague (1970) allow to build bridges with the semantics of formal languages.





A key milestone...

(SHRDLU, Winograd 1972)




```
HUMAN: Pick up a big red block.
SHRDLU: OK.
HUMAN: Grasp the pyramid.
SHRDLU: I don't understand which pyramid you mean
HUMAN: Find a block which is taller than the one you are holding
       and put it into the box.
SHRDLU: by "it", i assume you mean the block which is taller than the one
          I am holding.
SHRDLU: ok.
HUMAN: What does the box contain?
SHRDLU: The blue pyramid and the blue block.
```

The limits of symbolic AI

SHRDLU is one of the first impressive chatbots operating on a **closed world**. With few ambiguities and such a small world, it is possible to formalize a language to talk with the bot and the physics and implicits of the world.

The problem

SHRDLU does not generalize

Modeling language in an **open world** requires modeling a larger grammar but also world knowledge, common sense, wider discourse context...and it becomes pretty hard to do that with symbolic rules only.

Ambiguities

In an open world, ambiguities become one of the key problem:

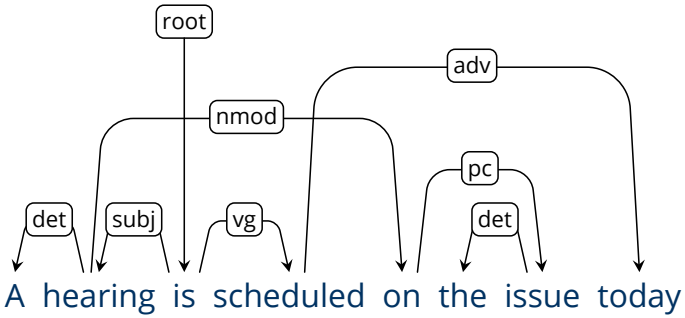
- Consider :
  1. Q1: John eats a salad with a knife
  2. Q2: John eats a salad with tomatoes
- With what instrument does John eat the salad ?
  1. A1 : with a knife
  2. A2 : with tomatoes (???)

Implicit knowledge

For humans the answer to Q1 is obvious, the answer to Q2 should be problematic. For a computer, predicting this contrast requires to model a database of world knowledge that is not reasonable (attempts have failed)

Statistical models of language structure

- **From the beginning of the 1990**, to address ambiguity problems, and with the augmentation of storage and computing power, the first machine learning models for natural language processing start to emerge.
- There is the key hypothesis that language has hidden **structure**. For instance:

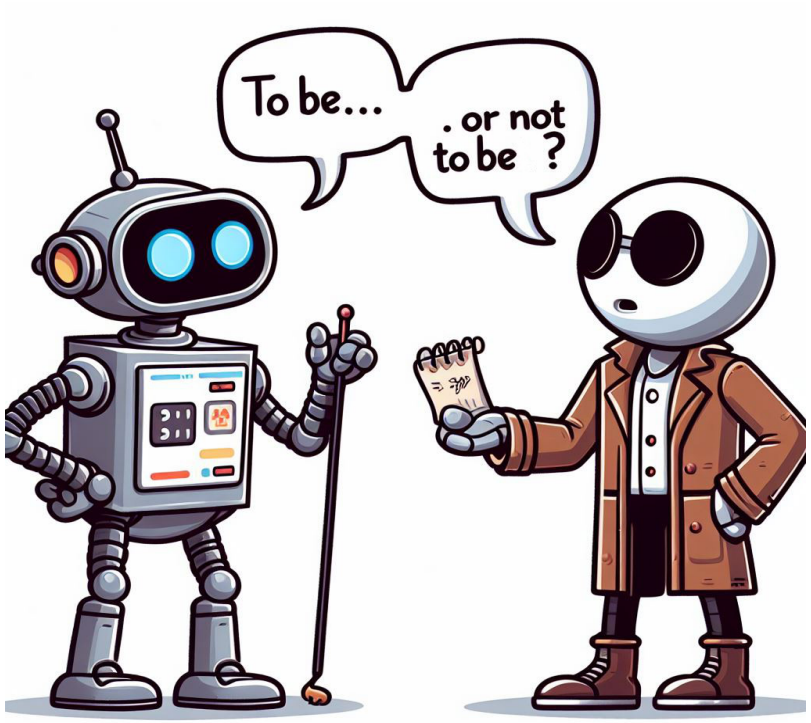


The problem

Generally speaking, annotated data for learning structured models of language is very costly/time consuming to get. Annoted corpora are generally small and not representative (natural language is far from being iid)

Outline

1. Early Machine Translation
2. Symbolic NLP
3. Language models
4. Deep learning language models



Classical statistics

The standard dataset in statistics gathers **measured data**

Example (Iris)

Sepal Length	Sepal Width	Petal Length	Petal Width	Species
5.1	3.5	1.4	0.2	setosa
4.9	3.0	1.4	0.2	setosa
4.7	3.2	1.3	0.2	setosa
4.6	3.1	1.5	0.2	setosa
5.0	3.6	1.4	0.2	setosa

- From there one can immediately use linear regression to perform **predictions**

$$y = \mathbf{w}^\top \mathbf{x} + b$$

for instance:

$$SepalLength = w_1 SepalWidth + w_2 PetalLength + w_3 PetalWidth + b$$

Machine learning for natural language

The standard dataset for written language is made of **symbolic data**

Example

token	pos	next token
the	D	cat
cat	N	sleeps
sleeps	V	on
on	P	the
the	D	mat
mat	N	[eos]

- Machine learning for natural language has to code symbols on mathematical objects (vectors) prior to learn a mathematical model
- The vectors coding the symbols are called **representations**

One hot representations

One hot encoding is the naive default representation. It amount to create a dictionary mapping symbols to vectors:

the	1	0	0	0	0	0
cat	0	1	0	0	0	0
sleeps	0	0	1	0	0	0
on	0	0	0	0	1	0
mat	0	0	0	0	0	1

- The input symbols are just replaced by their representation vector
- The statistical model outputs scores (real numbers) for each output symbol, thus the basic model becomes:

$$\mathrm{y} = \mathrm{W}\mathrm{x} + \mathrm{b}$$

where $\mathrm{y} \in \mathbb{R}^d$ has the size of the set of output symbols and W and b are the parameters

Turning scores into probabilities

- One can easily map a vector of **positive reals** to probabilities:

$$P(y_i) = \frac{y_i}{\sum_{j=1}^{d} y_j}$$

- In case the vector contains positive and negative reals, one uses the **softmax** function:

$$P(y_i) = \frac{\exp(y_i)}{\sum_{j=1}^{d} \exp(y_j)}$$

- The following model is called **softmax regression**, or multinomial logistic regression:

$$\mathrm{y} = \mathrm{softmax}(\mathrm{Wx} + \mathrm{b})$$

where $\mathrm{y} \in [0, 1]^d$ and $\sum_{j=1}^{d} y_j = 1$

Example

Next word prediction

How do we compute $P(Y = sleeps|cat)$?

1. We map *cat* to its representation x = (0,1,0,0,0,0):
2. We compute a score for each output word with softmax regression

$$\mathrm{y} = \mathrm{softmax}(\mathrm{W}\mathrm{x} + \mathrm{b})$$

3. We pick $P(Y = sleeps|cat)$ by looking up the probability of *sleep* at the relevant position in y

Where do the parameters come from ?

Recall that linear regression minimizes the Mean square error on a dataset to estimate the parameters, a softmax regression minimizes a **loss** function that plays the same role

Generative language model

- Let a sentence be the sequence of words $\omega_1 \dots \omega_n$, a **language model** naturally computes the probability of a sentence $P(\omega_1 \dots \omega_n)$ relying on the chain rule of probability:

$$P(\omega_1 \dots \omega_n) = \prod_{i=1}^{n} P(\omega_i | \omega_1 \dots \omega_{i-1})$$

- In practice one limits the context to some constant k (markov assumption)

$$P(\omega_1 \dots \omega_n) \approx \prod_{i=1}^{n} P(\omega_i | \omega_{i-k} \dots \omega_{i-1})$$

Example

A bigram language model is a language model where $k = 1$ and each word is predicted given its predecessor only

$$P(\text{the cat sleeps on the mat}) = P(cat|the) \times P(sleeps|cat) \times P(on|sleeps) \times P(the|on) \times P(mat|the)$$

Softmax language model of higher order

Here is how you increase the context size to get a trigram model:

prev token	token	pos	next token
[bos]	the	D	cat
the	cat	N	sleeps
cat	sleeps	V	on
sleeps	on	P	the
on	the	D	mat
the	mat	N	[eos]

The model computes sentence probabilities as :

$$P(\omega_1 \dots \omega_n) = \prod_{i=1}^{n} P(\omega_i | \omega_{i-2}, \omega_{i-1})$$

where:

$$P(\Omega | \omega_{i-2}, \omega_{i-1}) = \text{softmax}\left(\mathrm{W}\, \mathrm{x}_{\text{prev token}} \parallel \mathrm{x}_{\text{token}} + \mathrm{b}\right)$$

Importance of the context size

from Shakespeare works

The larger the context (the larger the *k*) the better the model. Here are examples of randomly generated sentences with increasing values of *k*:

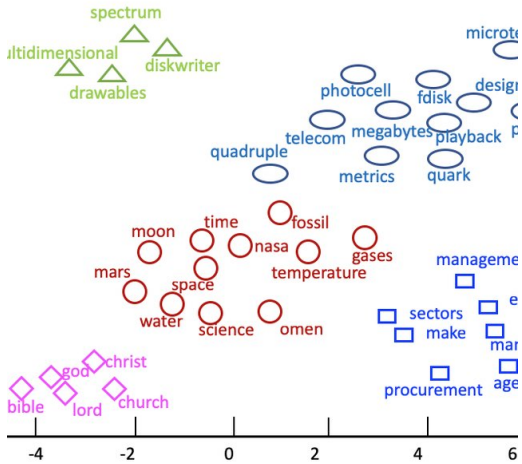
- **k=1**, ”care I of . its destined , in , from . . for the Felix an not which measure excited”
- **k=2**, ”I compassionated him , how heavily ; but I had arrived , but they averred , my unhallowed damps and”
- **k=3**, ”I continued their single offspring . At length I gathered from a man who , born in freedom , spurned”
- **k=4**, ”I continued walking in this manner , during which I enjoyed the feeling of happiness . Still thou canst listen”
- **k=5**, ”I continued walking in this manner for some time , and I feared the effects of the daemon’s disappointment .”

Word embeddings

Less naive word representations

A word embedding algorithm creates small dimensional vectors that are verify the distributional hypothesis

- One hot word vectors are highly dimensional (size of the vocabulary) and their geometry is uninteresting (they are all equally similar with cosine similarity).
- **Word embeddings** are small dimensional vectors where vectors of words that are semantically similar are similar (wrt cosine similarity)
- **Distributional hypothesis** Words that occur in similar textual contexts are semantically similar

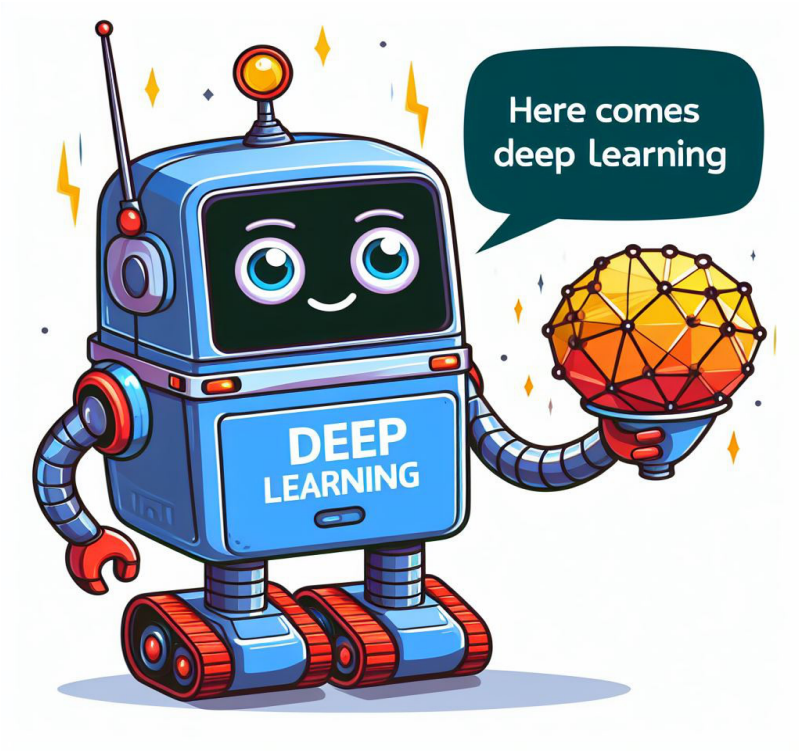


Relation with traditional methods

Other methods of dimensionality reduction were used before to compute word embeddings (Latent Semantic analysis) but had quadratic complexity in the number of examples. `word2vec` Mikolov (2013) **scales up** with linear complexity

Outline

1. Early Machine Translation
2. Symbolic NLP
3. Language models
4. Deep learning language models



Neural network language model

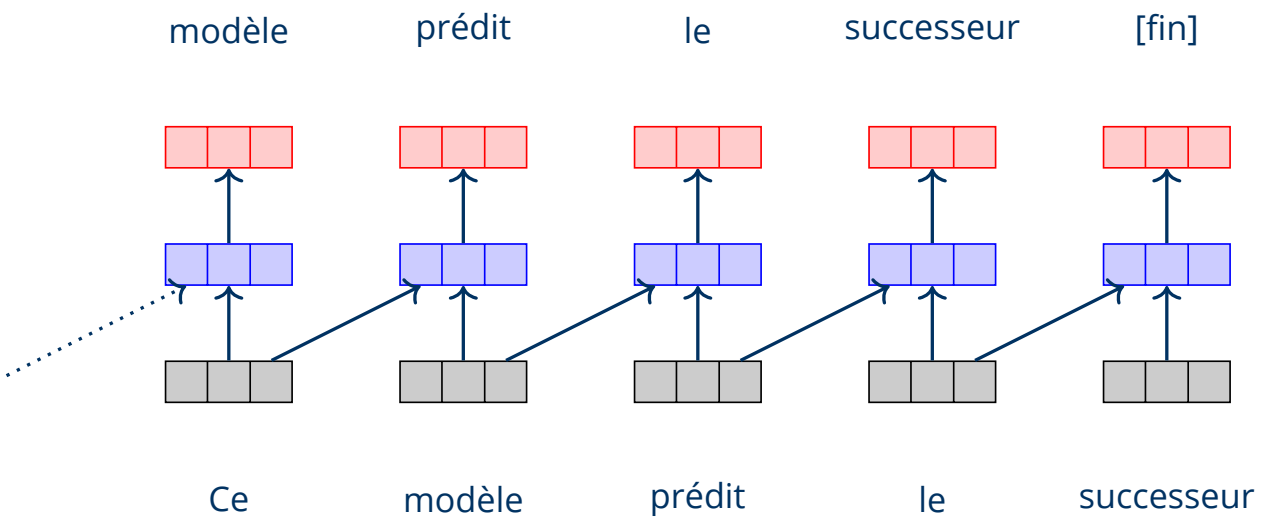
(Bengio et al. 2003)

A neural network language model maps word symbols to one vectors (x). The one hot vector is used to look up for the word embedding (e) into the matrix of word embeddings E and those are concatenated to get the context embedding h

$$P(Y|\omega_{i-k}\dots\omega_{i-1}) = \text{softmax}(\mathrm{Wh} + \mathrm{b})$$

$$\mathrm{h} = \mathrm{Ex}_{i-k} \parallel \dots \parallel \mathrm{Ex}_{i-1}$$

Example (Trigram NNLM)



Université Paris Cité

Recurrent neural language models

RNNLM (Mikolov et al. 2011)

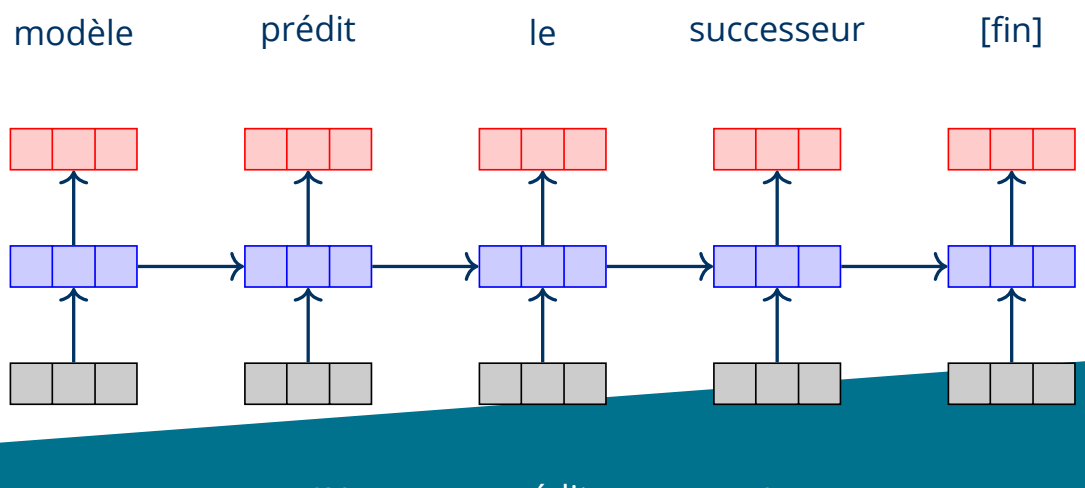
- **Longer contexts** Markovian hypotheses are relaxed:

$$P(\omega_1 \dots \omega_n) = \prod_{i=1}^{n} P(\omega_i | \omega_1 \dots \omega_{i-1})$$

- **Recurrent Neural Networks** (Elman 1991) are state-based models

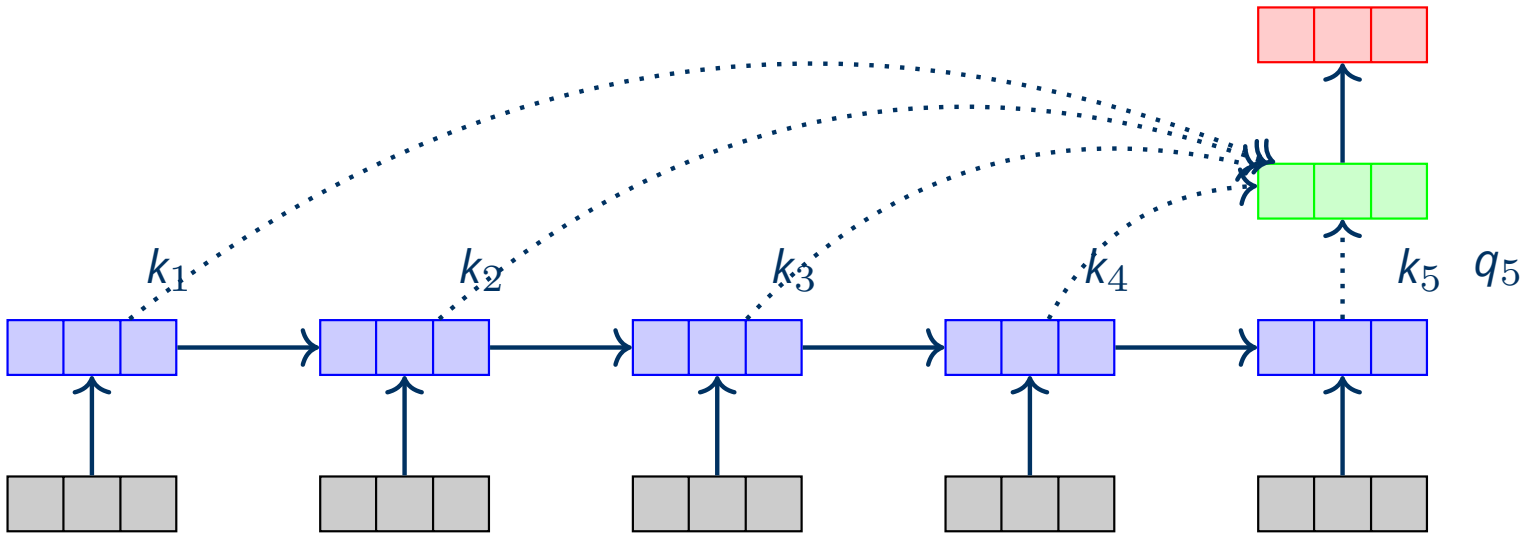
$$\mathbf{h}_t = \tanh(\mathbf{W}\mathbf{h}_{t-1} + \mathbf{U}\mathbf{e}_t + \mathbf{b})$$
$$\mathbf{e}_t = \mathbf{E}\mathbf{x}_t$$

- The conditional probabilities are now dependant of **contextualised embeddings**



Attention

- In practice RNN models (including LSTMs and GRU) can take only a finite context into account
- To correct the problem, Bahdanau (2014) introduces an **attention** module that allows to access past elements directly



$$a_{ij} = \mathrm{q}_i^\top \mathrm{k}_j$$

$$\alpha_i = \mathrm{softmax}(a_{i1} \ldots a_{ii})$$

Summary

- The first attempts in language modeling were statistically based (information theory).
- Problems of ambiguity, world knowledge and observations on the intrinsic structure of language motivated symbolic AI approaches
- Ambiguity problems have been largely adressed with ”classical” (convex) machine learning methods. The bottleneck is supervision since it requires human annotations
- Deep learning methods and LLMs manage to learn on tremendous amount of data because they are **self supervised** and they are able to capture very large contexts.
  - An LLM is learned on a very large amount of raw text (**foundational model**)
  - The downstream task is learned on a smaller annotated data set (**fine-tuning**)

The next step ?

RNNs as described so far are still relatively inefficient because they are state based. Transformer-based generative language models manage to **scale up** using a fully parallel model architecture.