Methods in computational linguistics

Neural word embeddings II

Dmitry Nikolaev, IMS, WS 2021/2022

We need embeddings for words in contexts

1. We need different embeddings for different contexts.
2. We do no want to ignore word order any more.

We need embeddings for words in contexts

Dealing with contextual variability:

The general idea: replace dictionary lookup (word2vec) with a pre-trained model.

Previously embeddings were computed once and then stored; now the embeddings are recomputed for each sentence.

We need embeddings for words in contexts

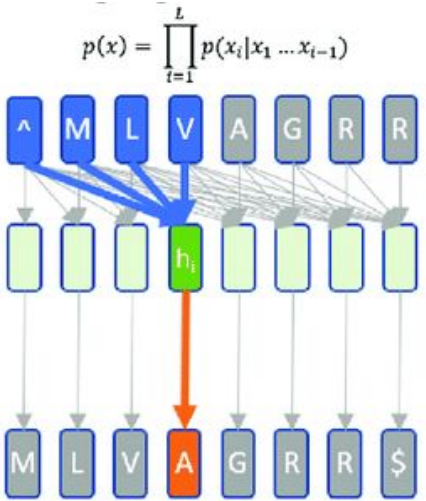
Dealing with word order:

We need to find a model should be able to incorporate information about word order.

What kind of model do we need?

But a language model, of course.
(More on LMs next Friday.)

Pre-neural language models computed the probability of the next token directly. Neural models need to compute the representation first.

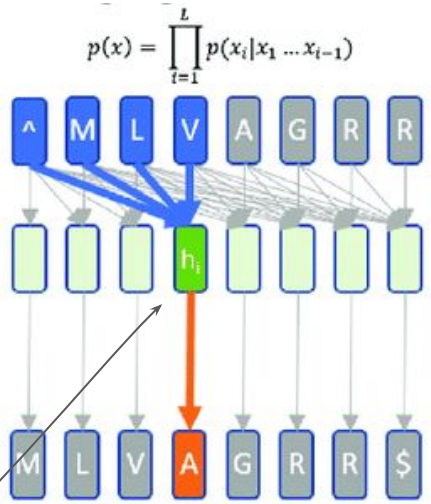


https://www.researchgate.net/figure/Diagram-of-model-architectures-and-language-modeling-approaches-A-Language-models-model_fig1_352468894

What kind of model do we need?

But a language model, of course.

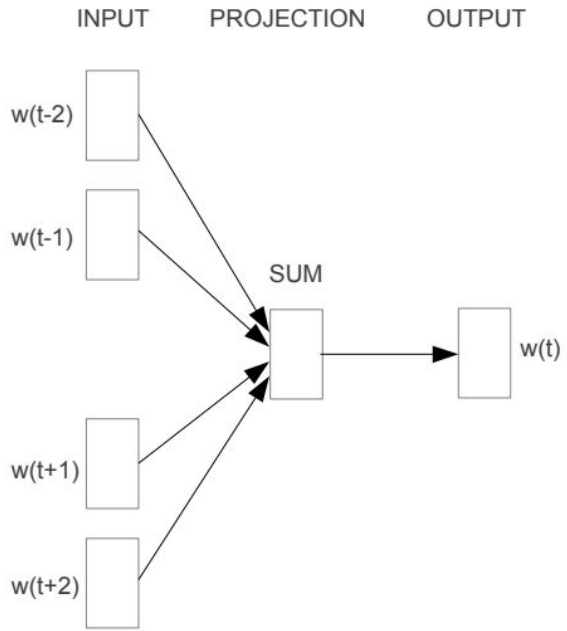
Pre-neural language models computed the probability of the next token directly. Neural models need to compute the representation first.



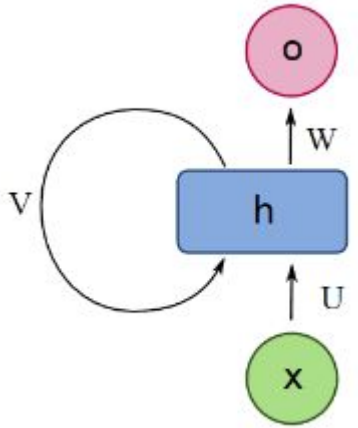
https://www.researchgate.net/figure/Diagram-of-model-architectures-and-language-modeling-approaches-A-Language-models-model_fig1_352468894

The classical neural architecture for sequences

Recurrent language models (RNNs):



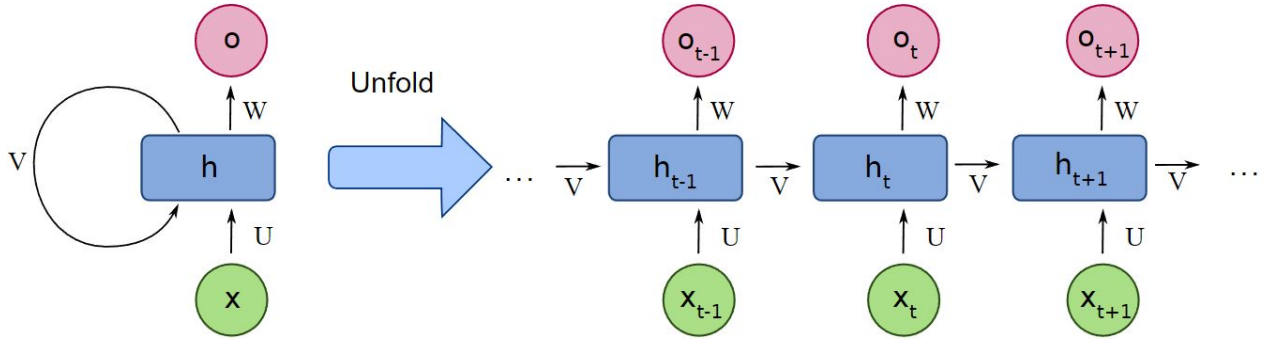
vs.



A bit of history

RNNs were first developed in the 1980s and first large implementations were proposed in early 1990s.

They were not practical because a copy of the whole networks had to be stored for each time step (e.g., each word) for the backward pass, so they had to be small:



Theory vs. practice

In theory, RNNs are superpowerful because they can process inputs of arbitrary length.

In practice, they failed for even relatively short sequences.

LSTM revolution

In 1997, a new type of RNN (LSTM) was proposed by Sepp Hochreiter and Jürgen Schmidhuber, which soon became a standard tool for sequence modelling.

It handled longer sequences much better.

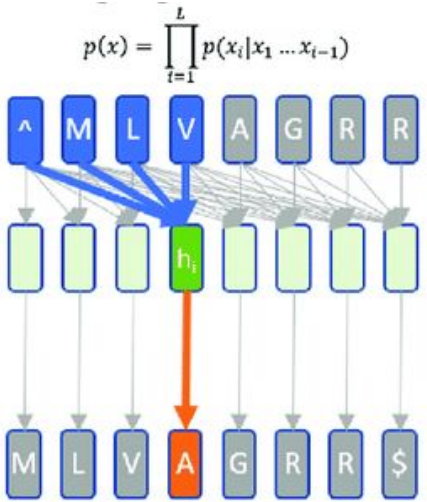
Sequence-to-sequence paradigm

With the availability of powerful RNNs, it became possible to reformulate many problems as sequence-to-sequence problems:

- Machine translation: *I went home* -> הלכתי הביתה
- POS tagging: *I went home* -> PRON VERB ADV
- Sentiment analysis: *I went home* -> Neutral
- Grammatical error correction: *I gone home* -> *I went home*

Straightforward approach to RNN embeddings

Try predicting next word from a prefix (beginning of a sentence / short text fragment) using an LSTM; take the hidden representation as the embedding.



Causal limitation

LSTM only goes left to right:

Quick _
Quick brown _
Quick brown fox _
Quick brown fox jumps _

In principle, it can handle ‘Quick brown fox _ over me.’ (read all the sequence, ask for identity of the missing token), but this wasn’t explored at the time.

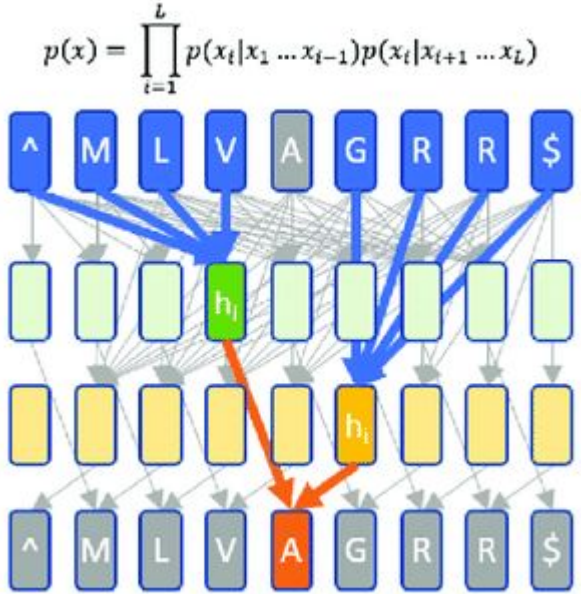
Causal limitation

The right context is also important for representations.

E.g., words often appearing at the beginning of a sentence will have poor representations.

A solution

Use two independent language models and combine (e.g., concatenate) the representations:



https://www.researchgate.net/figure/Diagram-of-model-architectures-and-language-modeling-approaches-A-Language-models-model_fig1_352468894

Another problem

Even LSTMs did a bad job of handling long-distance connections:

This is the method I've chosen to bend spoons with.

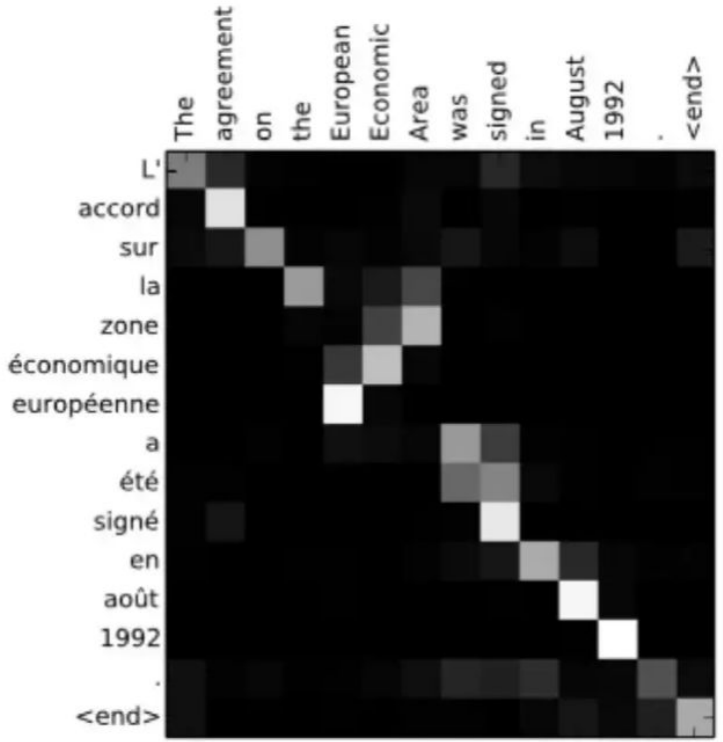
Dealing with long-distance connections

The second killer feature for RNNs (after the LSTM arch): *attention*.

The idea: at each step, when predicting the token, in addition to looking at the hidden state, look at the already computed embeddings of the prefix tokens and decide which one is the most relevant for the current token.

Attention

An idea from machine translation



NEURAL MACHINE TRANSLATION
BY JOINTLY LEARNING TO ALIGN AND TRANSLATE

Dzmitry Bahdanau
Jacobs University Bremen, Germany

KyungHyun Cho **Yoshua Bengio***
Université de Montréal

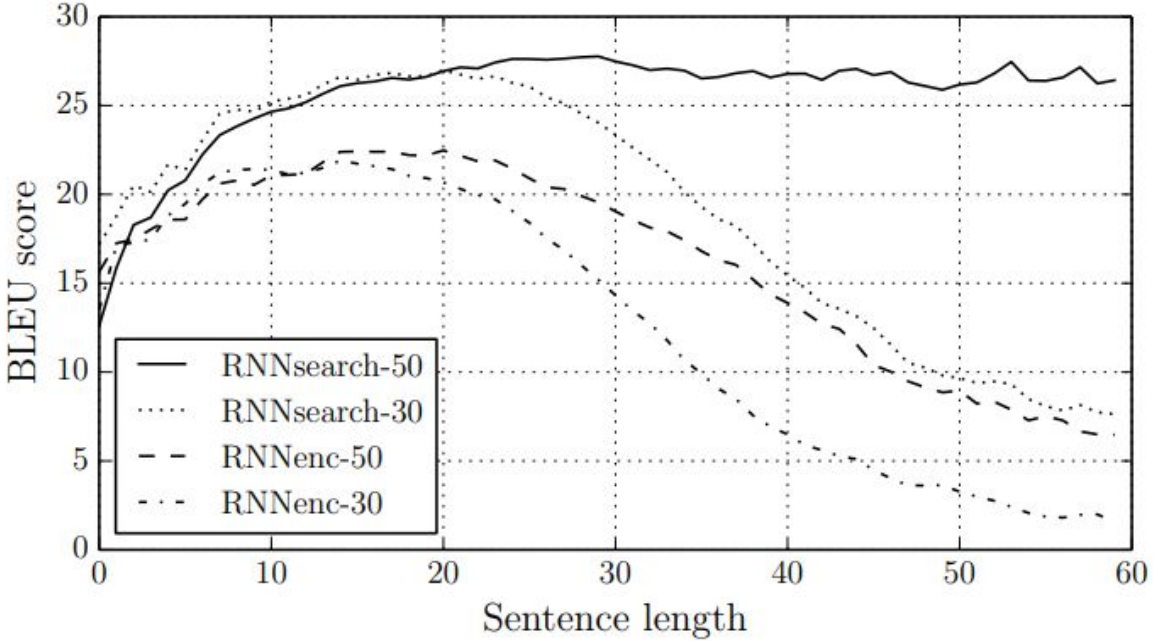


Figure 2: The BLEU scores of the generated translations on the test set with respect to the lengths of the sentences. The results are on the full test set which includes sentences having unknown words to the models.

Learned in Translation: Contextualized Word Vectors

Bryan McCann
bmccann@salesforce.com

James Bradbury
james.bradbury@salesforce.com

Caiming Xiong
cxiong@salesforce.com

Richard Socher
rsocher@salesforce.com

Abstract

Computer vision has benefited from initializing multiple deep layers with weights pretrained on large supervised training sets like ImageNet. Natural language processing (NLP) typically sees initialization of only the lowest layer of deep models with pretrained word vectors. In this paper, we use a deep LSTM encoder from an attentional sequence-to-sequence model trained for machine translation (MT) to contextualize word vectors. We show that adding these context vectors (CoVe) improves performance over using only unsupervised word and character vectors on a wide variety of common NLP tasks: sentiment analysis (SST, IMDb), question classification (TREC), entailment (SNLI), and question answering (SQuAD). For fine-grained sentiment analysis and entailment, CoVe improves performance of our baseline models to the state of the art.

What’s next? Add more layers!

Deep contextualized word representations

Matthew E. Peters[†], **Mark Neumann**[†], **Mohit Iyyer**[†], **Matt Gardner**[†],
{matthewp,markn,mohiti,mattg}@allenai.org

Christopher Clark[*], **Kenton Lee**[*], **Luke Zettlemoyer**[†*]
{csquared,kentonl,lsz}@cs.washington.edu

ELMo

‘We use vectors derived from a bidirectional LSTM that is trained with a *coupled language model (LM) objective* on a large text corpus. For this reason, we call them ELMo (Embeddings from Language Models) representations.’

Advantages

Can directly use the outputs for word-sense disambiguation, much better results on token classification.

	Source	Nearest Neighbors
GloVe	play	playing, game, games, played, players, plays, player, Play, football, multiplayer
biLM	Chico Ruiz made a spectacular play on Alusik 's grounder {...}	Kieffer , the only junior in the group , was commended for his ability to hit in the clutch , as well as his all-round excellent play .
	Olivia De Havilland signed to do a Broadway play for Garson {...}	{...} they were actors who had been handed fat roles in a successful play , and had talent enough to fill the roles competently , with nice understatement .

Table 4: Nearest neighbors to "play" using GloVe and the context embeddings from a biLM.

Disadvantages

Training RNNs is a very slow and difficult process. This limited the size of the models, and for good language understanding we need big models.

Another breakthrough: get rid of the RNN

Attention Is All You Need

Ashish Vaswani* Google Brain avaswani@google.com

Noam Shazeer* Google Brain noam@google.com

Niki Parmar* Google Research nikip@google.com

Jakob Uszkoreit* Google Research usz@google.com

Llion Jones* Google Research llion@google.com

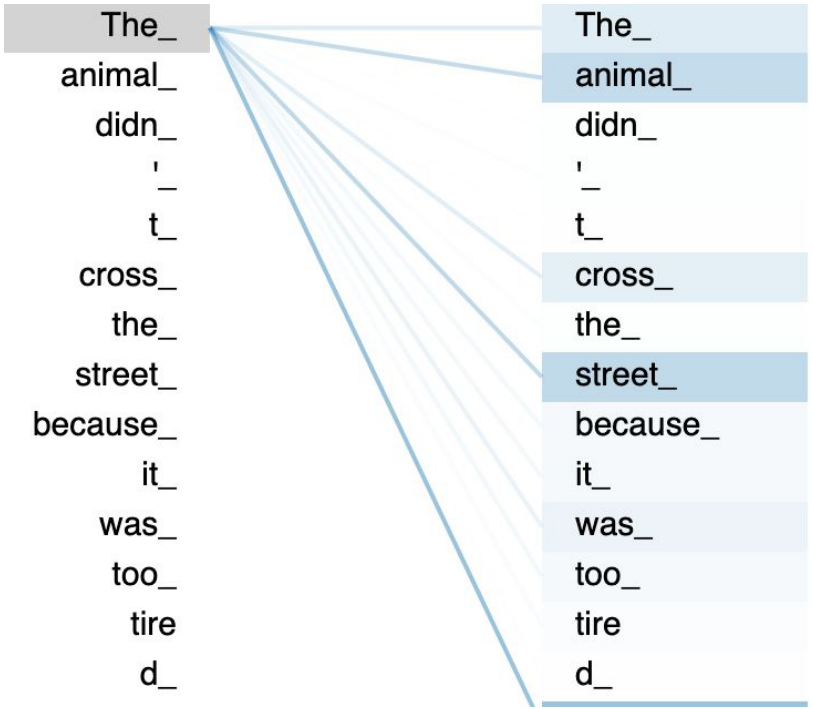
Aidan N. Gomez* † University of Toronto aidan@cs.toronto.edu

Łukasz Kaiser* Google Brain lukaszkaiser@google.com

Illia Polosukhin* ‡ illia.polosukhin@gmail.com

Transformer and *self-attention*

With self-attention any token can use information from any other token.



https://towardsdatascience.com/self-attention-5b95ea164f61

Transformer and *self-attention*

With self-attention any token can use information from any other token. No need to unroll (copy) the network over time; only vertical layers are needed.

Much faster training.

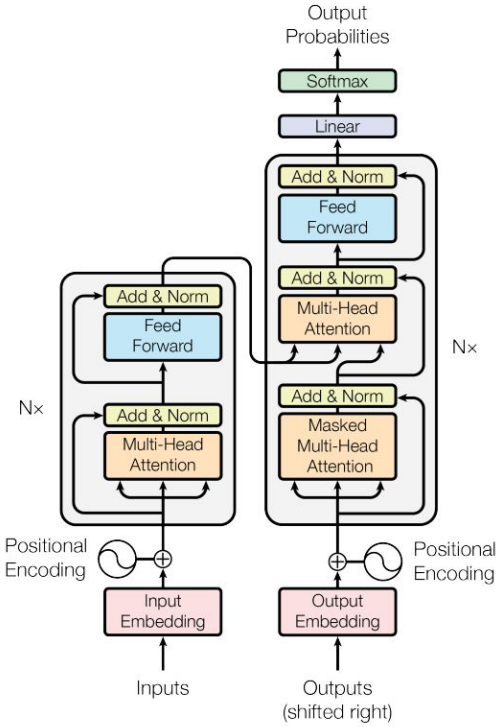
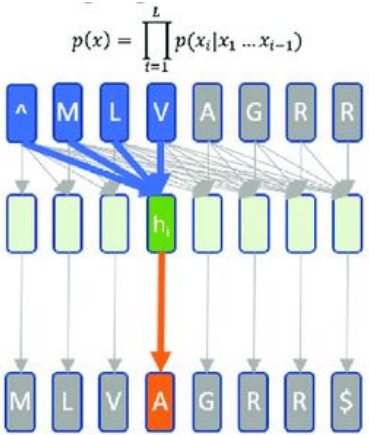


Figure 1: The Transformer - model architecture.

How to extract embeddings from this?

Language-model approach: hide the tokens to the right from the model (a.k.a. causal masking)



Output

<EOS>	1	1	1	1	1
lunch	1	1	1	1	0
eating	1	1	1	0	0
love	1	1	0	0	0
I	1	0	0	0	0

https://sshleifer.github.io/blog_v2/jupyter/2020/03/12/bart.html

We are still missing the right context.

We can concatenate together two self-attention-based models, but...

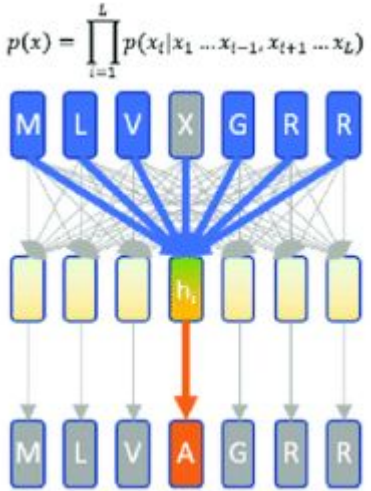
Self-attention treats all context tokens equally, so now we can do *bidirectional language modelling*:

_ brown fox jumps over me.
Quick _ fox jumps over me.
Quick brown fox _ over me.

Or even

Quick _ fox _ over me.

And process all this in parallel.



Masked language modelling

Bidirectional language modelling is also called *masked language modelling* because a special [MASK] token is used instead of _:

> Quick [MASK] fox jumps over me.

Cloze task

This is also called *cloze task* because it uses an old approach to language teaching:

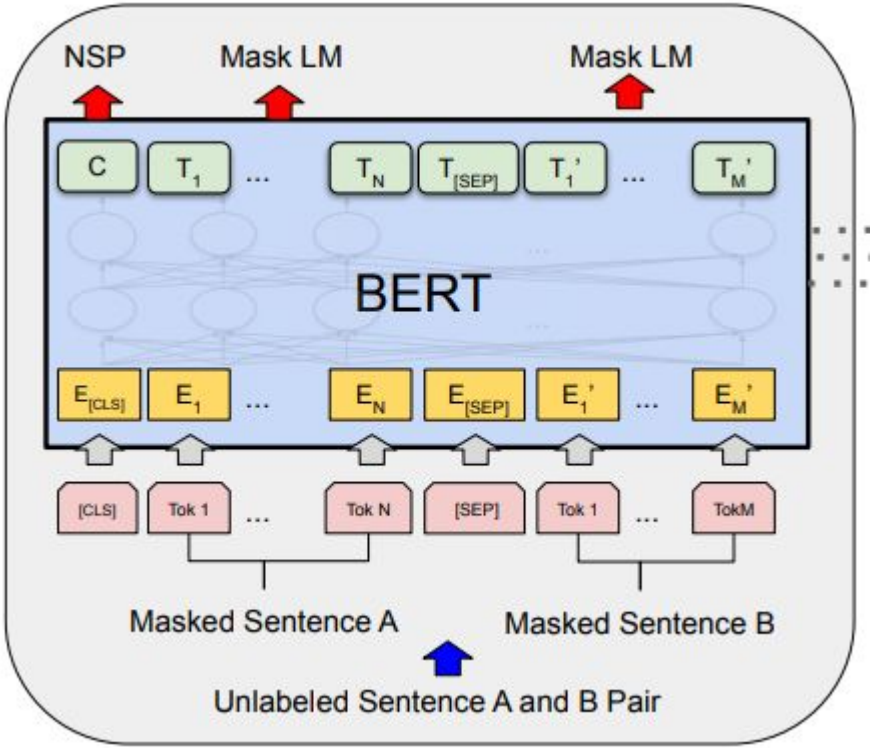
> Today, I went to the _____ and bought some milk and eggs. I knew it was going to rain, but I forgot to take my _____, and ended up getting wet on the way.

Enter BERT

BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

Jacob Devlin **Ming-Wei Chang** **Kenton Lee** **Kristina Toutanova**
Google AI Language
{jacobdevlin,mingweichang,kentonl,kristout}@google.com

BERT architecture



Bidirectional embeddings

Advantages:

- SOTA handling of long dependencies and token embeddings
- Fine-grained understanding of syntax
- Decent understanding of word meanings

Disadvantages:

- Sequence length is fixed in the architecture
- Self-attention demands a lot of parameters — an active area of research is how to save on those

The era of Transformers

Nearly all SOTA models in NLP are Transformer based.

A nice tutorial: https://nlp.seas.harvard.edu/2018/04/03/attention.html