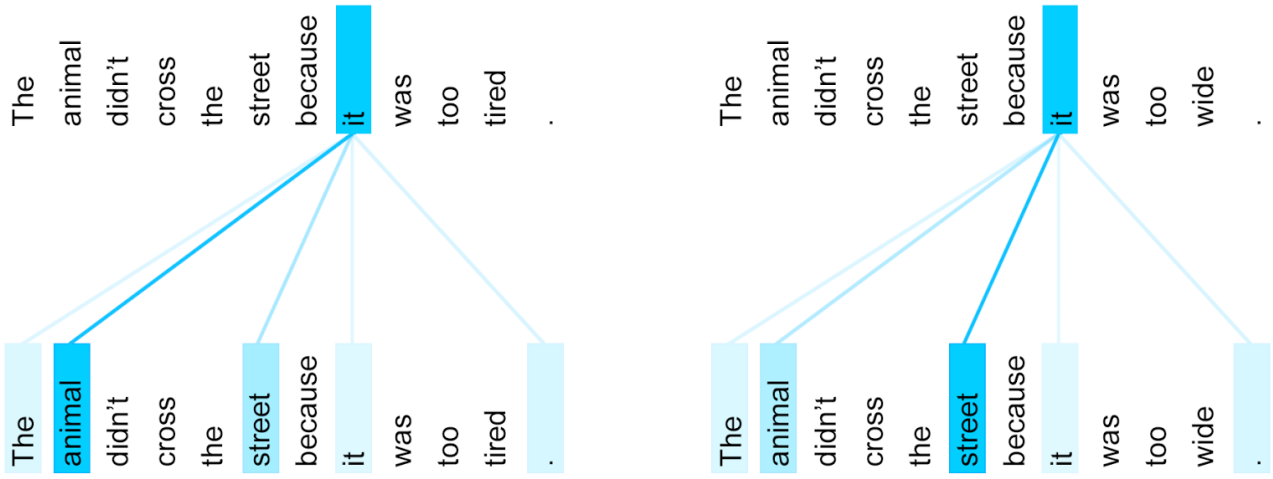
From Bag of Words to Transformers: 10 Years of Practical Natural Language Processing

Zelros AI
Nov 4 · 4 min read

A brief history of hands-on NLP through the lens of Kaggle.



Depending on the context (here, the last word of the sentence), 'it' can refer to 'animal' or 'street'. From Google introducing transformers.

Since its creation in 2010, **Kaggle**, the famous data science competition platform, has always been a great observatory of the evolution of machine learning trends. **Several generations of breakthroughs have been tested here**, engaging thousands of practitioners, and millions of forum discussions.

Among the various types of challenges issued on the online platform (computer vision, speech, tabular, …), **Natural Language Processing** (NLP) is today of particular interest. Indeed, this domains is witnessing several exciting innovations since a few months. The last one being the advent of **transformers and pretrained language models**.

NLP is one of the important technologies we are leveraging in our product at Zelros. That's why our research team is heavily investing in this domain. We love experimenting with new approaches, and share about what's happening in the NLP community. **Here is a short history of NLP technics viewed through the Kaggle platform lens**. We hope you will enjoy it! 😊

Until 2016: the reign of bag of words and TF-IDF

Before around 2016, the standard way of solving (and win! 🥇) Kaggle NLP challenges was to use **bag of words** (basically count how many times a word appears in a document) to create features feeding machine learning classifiers, like typically Naive Bayes. A slight refinement was **TF-IDF**.

This approach was used for example in the StumbleUpon Evergreen Classification Challenge (by the way, won by François Chollet in 2013 — yes the one who will create Keras two years after …).

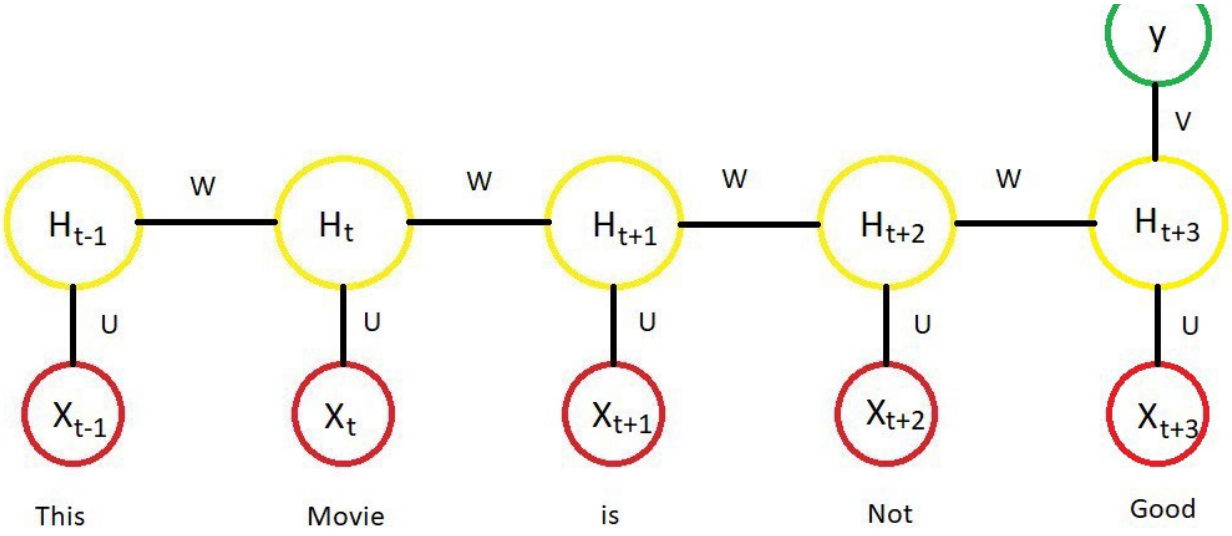
2016–2019: the rise of word embeddings + Keras and Tensorflow

In 2015, libraries for **dense word representations** appeared, such as **Gensim** (including **Word2vec** and **GloVe**). Other pretrained embeddings appeared afterwards, like Facebook **FastText** or **Paragram**.

At the same time, first versions of large audience, easy to use, **neural network frameworks** gained popularity — namely **Keras** and **Tensorflow**. With them it became possible to start capturing meaning in *sequences* of words, and no more only in *bags* of words.

To run deep neural networks, one last big hurdle was to tackle: **having access to high processing powers**. This was solved with the use of **low cost GPUs**. They became even more available with Kaggle providing them from free on their platform (through **collaborative Notebooks kernels**), after it has been acquired by Google in march 2017.

From this date, all the practical ingredients were present to make word embeddings and neural network (RNN, LSTM, GRU, … and refinements like attention) become **the standard way of solving NLP tasks on Kaggle**. So long TF-IDF…

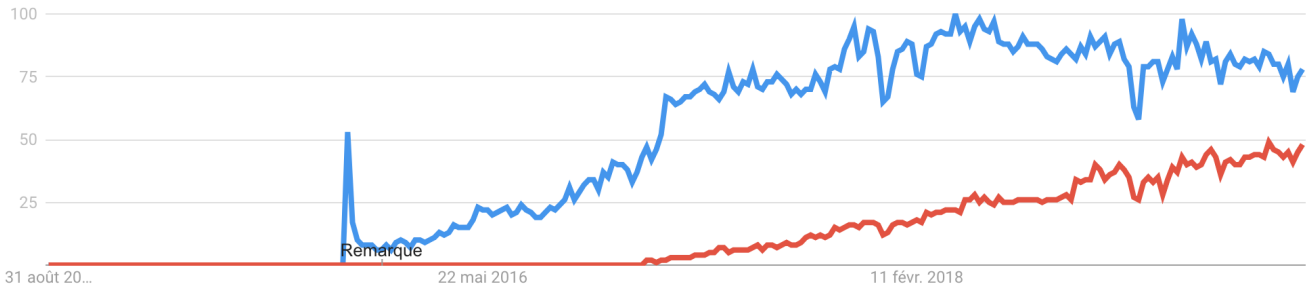


Encoding a sequence of words with a Recurrent Neural Network

2018–2019: the PyTorch breakthrough

Since a few months, **a new neural network framework is getting traction** among the data science community: **PyTorch**.

We won't enter into **Tensorflow VS PyTorch debate**, but what is sure is that an active community of PyTorch practitioners is growing on Kaggle. PyTorch notebooks and tutorials are regularly published on the platform.



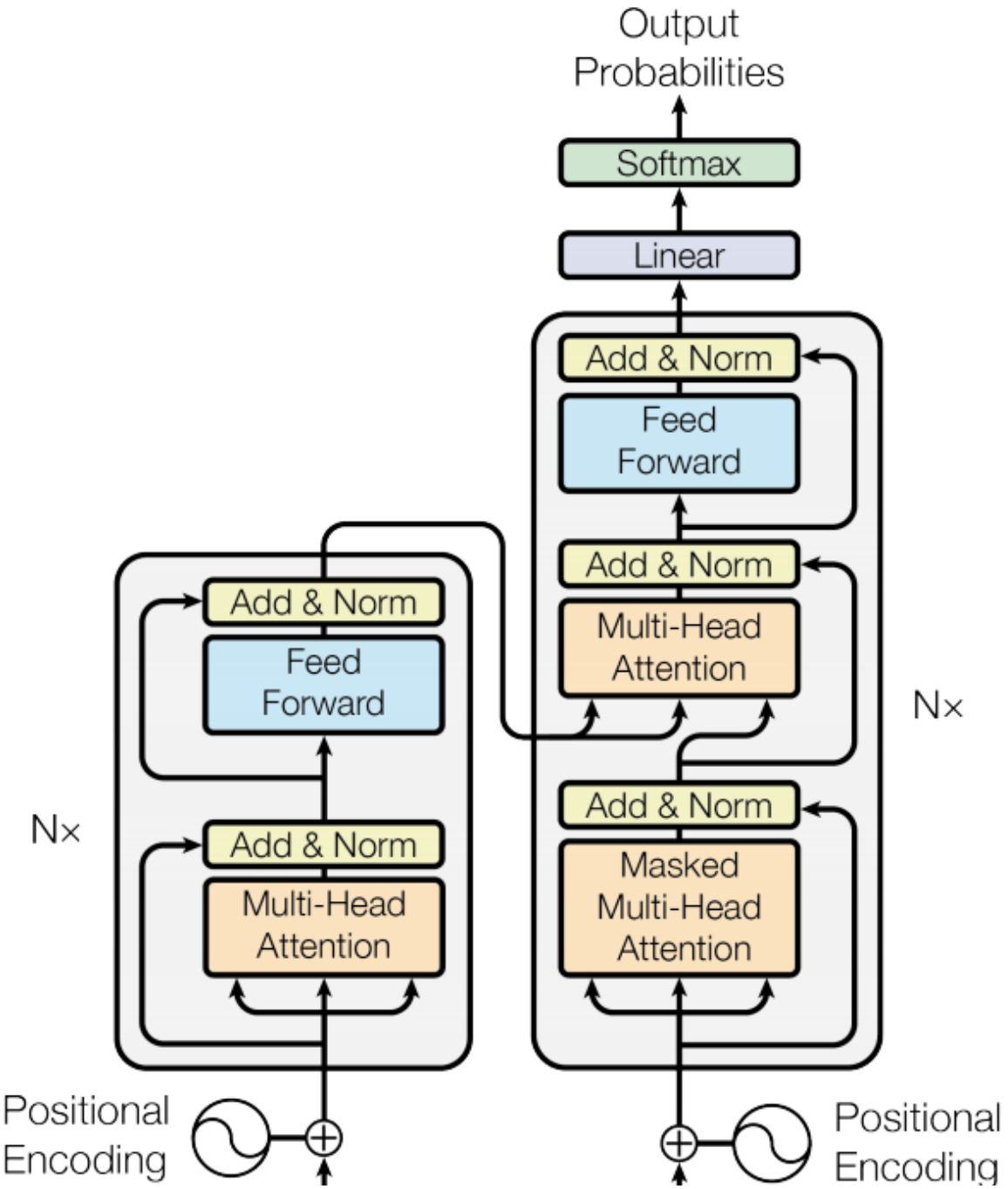
Tensorflow (blue) VS PyTorch (red) search trend in Google (source: Google Trend)

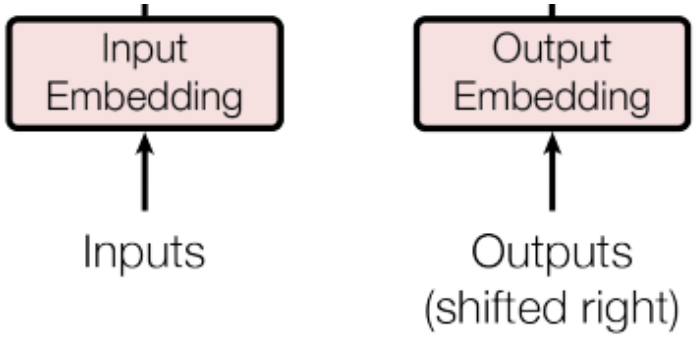
2019: the birth of transformers and pretrained language models

As seen in the previous sections, up to now the standard way of tacking NLP tasks was to use **word embeddings** (pretrained on large amounts of unlabeled data), use them to **initialize the first layer** of a neural network, and **train the other layers** on data of a particular task (may it be text classification, question answering, natural language inference, …).

If you think about it, **the problem is that this approach is not optimal**. Indeed, whenever you have a new task to solve, you have to learn again almost everything from zero. A model initialized with word embeddings always needs to **learn from scratch how to derive meaning from sequences of words** — although it is the core aspect of language understanding.

And here come **transformers**, the key paradigm shift that appeared in 2018: going from just initializing the first layer of models to **pretraining the entire models with hierarchical representations**. This opens new way of working: **transferring the information from a pretrained language model to downstream tasks** (a.k.a. transfer learning).





The Transformer — model architecture (from Attention Is All You Need paper)

In practice, today the best way to leverage pretrained language models is to use **the excellent transformers library from Hugging Face** (founded by french entrepreneurs now based in US, and alumni of Station F Microsoft AI Factory like us 🤗). It is now compatible with both PyTorch and TensorFlow. If you want a wrapper above it for simple tasks like text classification, you can **have a look at simple-transformers**.

And if you are focusing on non-english texts, **an other library to watch is fast.ai**, designed to incorporate pretrained models for different languages. It was created by Jeremy Howard, former President and Chief Scientist at… Kaggle :)

What's next?

Ready-to use libraries with last generation pretrained language models are now available for everyone. This allows **fast experimentation and a democratized usage of state-of-the art NLP technics**.

It will be interesting to follow how they will be used on **future Kaggle NLP competitions**. Like for example the recent **TensorFlow 2.0 Question Answering challenge**, to identify the answers to real user questions about Wikipedia page content. Stay tuned!

. . .

Want to join our Machine Learning team? Check-out our open positions!

Machine Learning NLP AI Deep Learning Neural Networks