Published in Towards Data Science

James Briggs Follow

Mar 10, 2021 · 10 min read · ✨ · Listen

HANDS-ON TUTORIALS

How to Apply Transformers to Any Length of Text

Restore the power of NLP for long sequences

Photo by Sebastian Staines on Unsplash

The de-facto standard in many natural language processing (NLP) tasks nowadays is to use a transformer. Text generation? *Transformer*. Question-and-answering? *Transformer*. Language classification? *Transformer*!

However, one of the problems with many of these models (a problem that is not just restricted to transformer models) is that we **cannot** process long pieces of text.

Almost every article I write on Medium contains 1000+ words, which, when tokenized for a transformer model like BERT, will produce 1000+ tokens. BERT (and many other transformer models) will consume **512 tokens max** — truncating anything beyond this length.

High-Level Approach

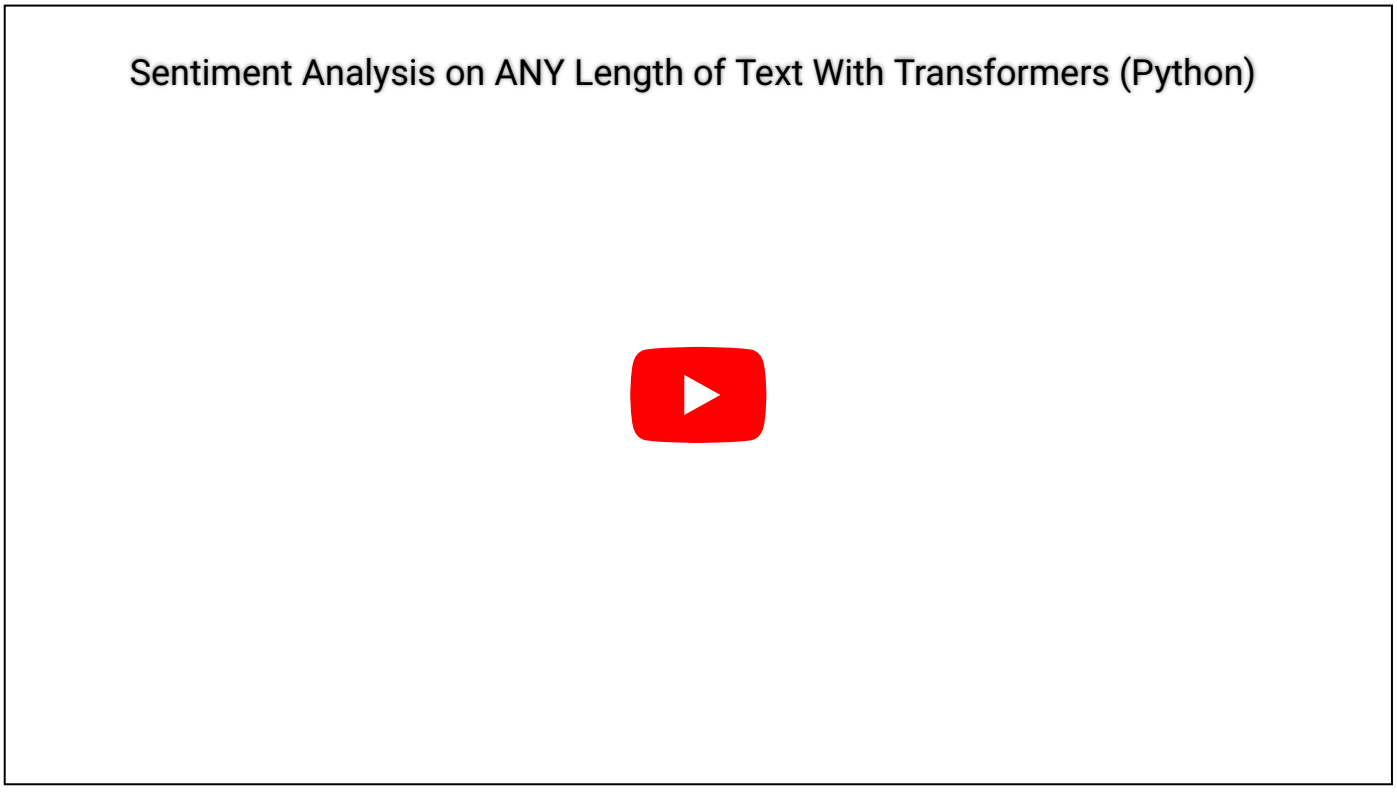
Getting Started
- Data
- Initialization

Tokenization

Preparing The Chunks
- Split
- CLS and SEP
- Padding
- Reshaping For BERT

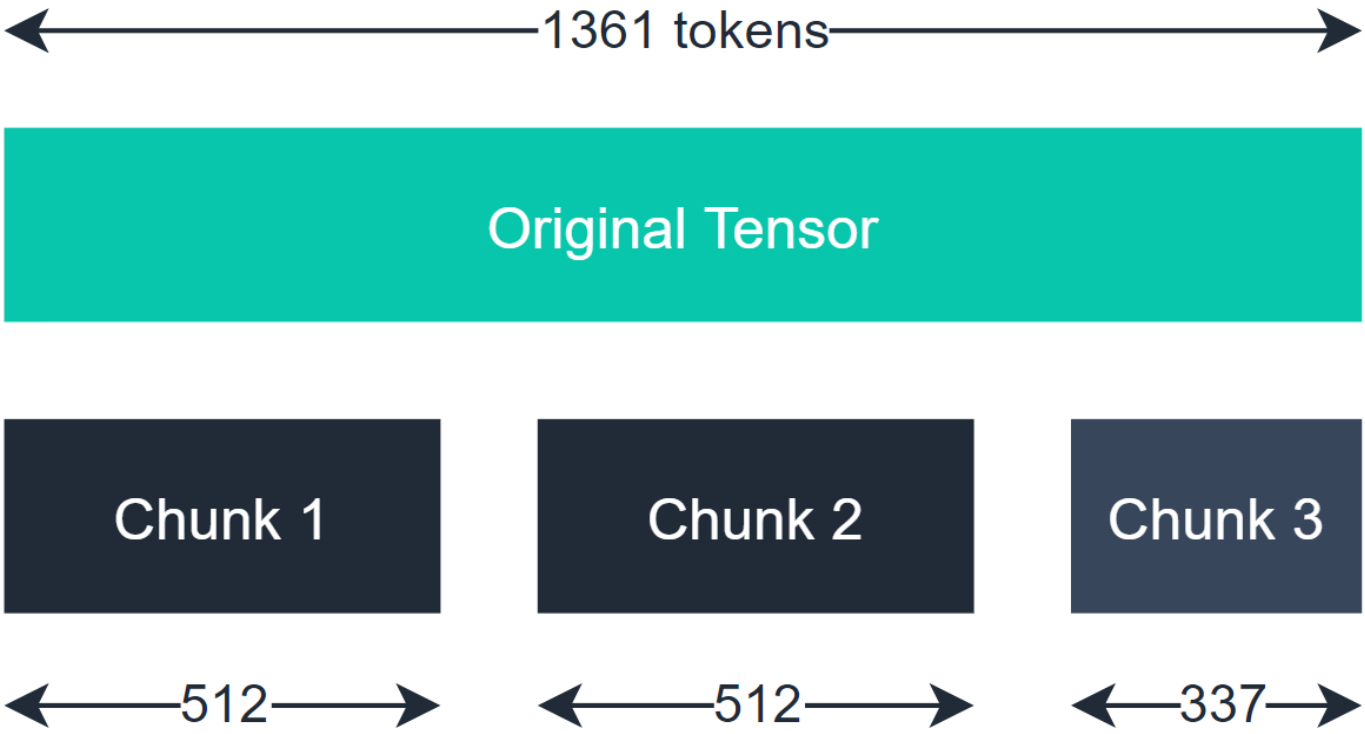
Making Predictions

If you prefer video, I cover everything here too:



simple.

We will be taking our text (say 1361 tokens) and breaking it into chunks containing no more than 512 tokens each.



A tensor containing 1361 tokens can be split into three smaller tensors. The first two would contain 512 tokens each, with the final tensor containing the remaining 337 tokens.

Once we have our chunks and transformed them so that they are ready to be consumed by BERT (more on that soon) — we pass them through our model and retrieve the sentiment scores for each chunk.

Finally, an average is taken for each sentiment class — providing us with an overall sentiment prediction for the entire piece of text (all 1361 tokens).

Now, explaining the high-level approach is one-thing. Writing it out is another. So let's start work through an example.

Data

First, we need some data to process. I found this rather long post on /r/investing:

```
I would like to get your all  thoughts on the bond yield increase this
week.  I am not worried about the market downturn but the sudden
increase in yields. On 2/16 the 10 year bonds yields increased by
almost  9 percent and on 2/19 the yield increased by almost 5 percent.

Key Points from the CNBC Article:

* **The “taper tantrum” in 2013 was a sudden spike in Treasury yields
due to market panic after the Federal Reserve announced that it would
begin tapering its quantitative easing program.**
* **Major central banks around the world have cut interest rates to
historic lows and launched unprecedented quantities of asset purchases
in a bid to shore up the economy throughout the pandemic.**
* **However, the recent rise in yields suggests that some investors
are starting to anticipate a tightening of policy sooner than
anticipated to accommodate a potential rise in inflation.**

The recent rise in bond yields and U.S. inflation expectations has
some investors wary that a repeat of the 2013 “taper tantrum” could be
on the horizon.

The benchmark U.S. 10-year Treasury note climbed above 1.3% for the
first time since February 2020 earlier this week, while the 30-year
bond also hit its highest level for a year. Yields move inversely to
bond prices.

Yields tend to rise in lockstep with inflation expectations, which
have reached their highest levels in a decade in the U.S., powered by
increased prospects of a large fiscal stimulus package, progress on
vaccine rollouts and pent-up consumer demand.

The “taper tantrum” in 2013 was a sudden spike in Treasury yields due
to market panic after the Federal Reserve announced that it would
begin tapering its quantitative easing program.

Major central banks around the world have cut interest rates to
historic lows and launched unprecedented quantities of asset purchases
in a bid to shore up the economy throughout the pandemic. The Fed and
others have maintained supportive tones in recent policy meetings,
vowing to keep financial conditions loose as the global economy looks
to emerge from the Covid-19 pandemic.
```

With central bank support removed, bonds usually fall in price which sends yields higher. This can also spill over into stock markets as higher interest rates means more debt servicing for firms, causing traders to reassess the investing environment.

“The supportive stance from policymakers will likely remain in place until the vaccines have paved a way to some return to normality,” said Shane Balkham, chief investment officer at Beaufort Investment, in a research note this week.

“However, there will be a risk of another ‘taper tantrum’ similar to the one we witnessed in 2013, and this is our main focus for 2021,” Balkham projected, should policymakers begin to unwind this stimulus.

Long-term bond yields in Japan and Europe followed U.S. Treasurys higher toward the end of the week as bondholders shifted their portfolios.

“The fear is that these assets are priced to perfection when the ECB and Fed might eventually taper,” said Sebastien Galy, senior macro strategist at Nordea Asset Management, in a research note entitled “Little taper tantrum.”

“The odds of tapering are helped in the United States by better retail sales after four months of disappointment and the expectation of large issuance from the $1.9 trillion fiscal package.”

Galy suggested the Fed would likely extend the duration on its asset purchases, moderating the upward momentum in inflation.

“Equity markets have reacted negatively to higher yield as it offers an alternative to the dividend yield and a higher discount to long-term cash flows, making them focus more on medium-term growth such as cyclicals” he said. Cyclicals are stocks whose performance tends to align with economic cycles.

Galy expects this process to be more marked in the second half of the year when economic growth picks up, increasing the potential for tapering.

Tapering in the U.S., but not Europe

Allianz CEO Oliver Bäte told CNBC on Friday that there was a geographical divergence in how the German insurer is thinking about the prospect of interest rate hikes.



```
Bäte said, adding that the situation is different stateside.

“Because of the massive programs that have happened, the stimulus that
is happening, the dollar being the world’s reserve currency, there is
clearly a trend to stoke inflation and it is going to come. Again, I
don’t know when and how, but the interest rates have been steepening
and they should be steepening further.”

## Rising yields a ‘normal feature’

However, not all analysts are convinced that the rise in bond yields
is material for markets. In a note Friday, Barclays Head of European
Equity Strategy Emmanuel Cau suggested that rising bond yields were
overdue, as they had been lagging the improving macroeconomic outlook
for the second half of 2021, and said they were a “normal feature” of
economic recovery.

“With the key drivers of inflation pointing up, the prospect of even
more fiscal stimulus in the U.S. and pent up demand propelled by high
excess savings, it seems right for bond yields to catch-up with other
more advanced reflation trades,” Cau said, adding that central banks
remain “firmly on hold” given the balance of risks.

He argued that the steepening yield curve is “typical at the early
stages of the cycle,” and that so long as vaccine rollouts are
successful, growth continues to tick upward and central banks remain
cautious, reflationary moves across asset classes look “justified” and
equities should be able to withstand higher rates.

“Of course, after the strong move of the last few weeks, equities
could mark a pause as many sectors that have rallied with yields look
overbought, like commodities and banks,” Cau said.

“But at this stage, we think rising yields are more a confirmation of
the equity bull market than a threat, so dips should continue to be
bought.”
```

We will use this for our example.

Initialization

The next thing we need to do is initialize our model and tokenizer. We’re going to be using PyTorch and the HuggingFace transformers library for everything.



tokenizer, so we write:

```
In [2]:
from transformers import BertForSequenceClassification, BertTokenizer
import torch

tokenizer = BertTokenizer.from_pretrained('ProsusAI/finbert')
model = BertForSequenceClassification.from_pretrained('ProsusAI/finbert
```

finbert_init.ipynb hosted with ❤️ by GitHub view raw

Because we're working with finance-heavy language, we have loaded the `ProsusAI/finbert` model — a more finance savvy BERT [1]. You can find the model details here.

Tokenization

Tokenization is the process of converting a string of text into a list of tokens (individual words/punctuation) and/or token IDs (integers that map a word to a vector



```
txt = "<this is the large post included above>"

tokens = tokenizer.encode_plus(
    txt, add_special_tokens=True,
    max_length=512, truncation=True,
    padding="max_length"
)
```

Here we are using the tokenizers `encode_plus` method to create our tokens from the `txt` string.

- `add_special_tokens=True` adds special BERT tokens like *[CLS]*, *[SEP]*, and *[PAD]* to our new 'tokenized' encodings.
- `max_length=512` tells the encoder the target length of our encodings.
- `truncation=True` ensures we cut any sequences that are longer than the specified `max_length`.
- `padding="max_length"` tells the encoder to pad any sequences that are shorter than the `max_length` with padding tokens.

These parameters make up the typical approach to tokenization. Still, as you can see, they're simply not compatible when we are aiming to split a longer sequence into multiple shorter chunks.

For that, we modify the `encode_plus` method to not perform any truncation or padding.

Additionally, the special tokens *[CLS]* and *[SEP]* are expected at the start and end of a sequence, respectively. As we will be creating these sequences separately, we must also add these tokens separately too.

The new `encode_plus` method looks like this:



```
Out[3]: Token indices sequence length is longer than the specified maximum sequ
        ence length for this model (1345 > 512). Running this sequence through
        the model will result in indexing errors
        1345
        {'input_ids': tensor([[1045, 2052, 2066,  ..., 4149, 1012, 1524]]), 'to
        ken_type_ids': tensor([[0, 0, 0,  ..., 0, 0, 0]]), 'attention_mask': te
        nsor([[1, 1, 1,  ..., 1, 1, 1]])}
```

tokenizer_long.ipynb hosted with ❤️ by GitHub view raw

Which will return a dictionary containing three key-value pairs, `input_ids`, `token_type_ids`, and `attention_mask`.

We have *also* added `return_tensors='pt'` to return PyTorch tensors from the tokenizer (rather than Python lists).

Preparing The Chunks

Now we have our tokenized tensor; we need to break it into chunks of no more than **510** tokens. We choose 510 rather than 512 to leave two places spare to add our *[CLS]* and *[SEP]* tokens.

Split

We apply the `split` method to both our input IDs and attention mask tensors (we don't need the token type IDs and can discard them).



```
for tensor in input_id_chunks:
    print(len(tensor))
```

```
510
510
325
```

split_tokens.ipynb hosted with ❤️ by GitHub view raw

Now we have three chunks for each tensor set. Note that we will need to add padding to the final chunk as it will not satisfy the tensor size of 512 required by BERT.

CLS and SEP

Next, we add the start of sequence *[CLS]* and separator *[SEP]* tokens. For this, we can use the `torch.cat` function, which con**cat**enates a list of tensors.

Our tokens are already in token ID format, so we can refer to the special tokens table above to create the token ID versions of our *[CLS]* and *[SEP]* tokens.

Because we are doing this for multiple tensors, we place the `torch.cat` function into a for-loop and perform the concatenation for each of our chunks individually.

Additionally, our attention mask chunks are concatenated with **1**s rather than **101** and **102**. We do this because the attention mask does not contain *token IDs* but instead a set of **1**s and **0**s.

Zeros in the attention mask represent the location of padding tokens (which we will add next), and as *[CLS]* and *[SEP]* are not padding tokens, they are represented with **1**s.

Padding

We need to add padding to our tensor chunks to ensure they satisfy the 512 tensor length required by BERT.

Our first two chunks don't require any padding as they already satisfy this length



function.

We should add this statement to the same for-loop where we add our *[CLS]* and *[SEP]* tokens — if you need help with this, I've included the full scripts at the end of the article.

Stacking all of the tensors together is done using the `torch.stack` function.

We then format these into an input dictionary and change the input IDs tensor datatype to `long`, and the attention mask tensor datatype to `int` — as required by BERT.



Making Predictions

Making our predictions is the easy part. We pass our `input_dict` as a `**kwargs` argument to our `model` — ***kwargs* allows the model to match `input_ids` and `attention_mask` keywords to variables within the model.



probabilities, we must apply a softmax function to the output tensor.

Finally, we take the `mean` of the values in each class (or column) to get our final positive, negative, or neutral sentiment probability.



If you'd like to extract the winning class, we can add an `argmax` function:

There we have our sentiment predictions for longer pieces of text!

We've taken a long piece of text containing 1000s of tokens, broke it down into chunks, manually added special tokens, and calculated the average sentiment across all chunks.

More often than not, looking at the full-length of a text is absolutely required to understand the sentiment of the topic being discussed. We have built a method to make that possible and allow us to work around typical text size limitations.

If you'd like to see the code in full — you can find it in the references below (there are two notebooks, but number **two** contains the exact code used here).

I hope you've enjoyed the article. Let me know if you have any questions or suggestions via Twitter or in the comments below! If you're interested in more content like this, I



References

[1] D. Araci, FinBERT: Financial Sentiment Analysis with Pre-trained Language Models (2019)

Jupyter Notebook 1

Jupyter Notebook 2

🤖 NLP With Transformers Course

If you'd like to learn more about sentiment analysis with transformers (this time with TensorFlow), check out my article on language classification here:

Build a Natural Language Classifier With Bert and Tensorflow
Apply cutting-edge transformer models to your language problems
betterprogramming.pub