How to evaluate content quality with BERT

Marie Haynes recently had a really insightful podcast interview with John Muller.

I specifically enjoyed the conversation about BERT and its potential for content quality evaluation.

“M 26:40 – .. Is Google using BERT now to better understand now whether content is good?”

“J 27:00 – … It’s not so much to understand the quality of the content but more to understand what is this content about, what is this sentence about, what is this query about …”

Google has repeatedly said that it helps understand natural language better. Content quality assessment like humans do is still fairly complicated for machines to do.

“M 28:54 – … could Google treat that as a negative to say ‘oh this page looks like it was SEO-ed, these keywords are here for Google and make that an actual detriment to the page”

“J 29:41 – … they’re just adding thousands of variations of the same keywords to a page and then our keyword stuffing algorithm might kick in and say well actually this looks like keyword stuffing, and then our keyword stuffing algorithm might kick in …”

On the other hand, keywords stuffing is something that is easier for machines to spot. One way to check that is to see if the text is written in a nonsensical way.

“J 29:41 – … But I guess with regards to BERT one of the things that that could be done because a lot of these algorithms are open-sourced, there’s a lot of documentation and reference material around them, is to **try things out and**

to take some of this SEO text and throw it into one of these algorithms and see does the primary content get pulled out, are the entities able to be recognized properly and it's not one to one the same to how we would do it because I'm pretty sure our algorithms are based on similar ideas but probably tuned differently but it can give you some insight into is this written in such a way that it's actually too confusing for a system to understand what it is that they're writing about."

This is the part that got me excited. Trying this out is a great idea and precisely what we will do in this article.

Britney Muller from Moz shared a really good idea and Python notebook with the code to test it.

> *Hmm, but isn't that measuring 'quality' in a way?*
>
> *There should be clarity around how we speak about BERT(on its own or w/o fine tuning it's limited)*
>
> *BUT with other libraries like (CoLA), BERT can label whether or not a sentence is grammatically correct: https://t.co/txUud2be3H*
>
> *— Britney Muller (@BritneyMuller) June 4, 2020*

We can use BERT fine tuned on The Corpus of Linguistic Acceptability (CoLA) dataset for single sentence classification.

This model can help us determine which sentences are grammatically correct and which aren't. It could be used as one of several proxies for content quality.

It is obviously not foolproof, but can get us in the right direction.

Fine tuning BERT on CoLA

The Colab notebook linked in Britney's tweet is far too advanced for non-experts, so we are going to take a drastic shortcut!

We are going to use Ludwig, a very powerful and code-free deep learning toolkit from Uber, to do the same.

Here are the technical steps:

1. Fetch a target page and extract the text.
2. Split it into sentences.
3. Use our model to predict whether each sentence is grammatically correct or not.
4. Calculate and report grammatically correct and incorrect sentences

First, let's build our predictive model.

I coded a simple to follow Google Colab notebook with all the steps.

Copy the notebook to your Google Drive and change the runtime type to GPU.

You can use the form at the top to test the code on your own articles. You might need to change the CSS selector to extract relevant text per target page. The one included works with SEL articles.

You should be able to run all the cells (one at a time) and see the evaluation in action.

Building the predictive model

When you compare the original notebook to the one that I created, you will find that we avoided having to write a lot of advanced deep learning code.

In order to create our cutting edge model with Ludwig, we need to complete four simple steps:

1. Download and uncompress the CoLA dataset
2. Create the Ludwig model definition with the appropriate settings and hyper parameters
3. Run Ludwig to train the model
4. Evaluate the model with held back data in the CoLA dataset

You should be able to follow each of these steps in the notebook. I will explain my choices here and some of the nuances needed to make it work.

Google Colab comes with Tensorflow version 2.0 preinstalled, which is the latest version. But, Ludwig requires version 1.15.3.

Another important step is that you need to set up the GPU version of Tensorflow to finish the training quickly.

We accomplish this with the next few lines of code:

```
!pip install tensorflow-gpu==1.15.3
```

```
%tensorflow_version 1.x
import tensorflow as tf; print(tf.__version__)
```

After this, you need to restart the runtime using the menu item: *Runtime > Restart runtime*.

Run the form again, the line that imports pandas and continue to the step where you need to install Ludwig.

The accuracy of the predictive model can vary widely and it is heavily influenced by your choice of hyper parameters.

These are generally determined empirically by trial and error and to save time, I simply borrowed the ones from the weights and biases notebook.

As you can see above, in their visualization, the best combination results in a **validation accuracy of 84%.**

We added the same parameters to our model definition under the training section.

training:

batch_size: 16
learning_rate: 0.00003
epochs: 3

Next, we can train our BERT model on the CoLA dataset using a single command line.

```
!ludwig experiment --data_csv cola_dataset.csv --model_definition_file model_definition.yaml
```

We achieve a **validation accuracy of 80%**, which is slightly lower than the original notebook, but we put in significantly less effort!

Now, we have a powerful model that can classify sentences as grammatically correct or not.

I added additional code to the notebook to evaluate some test sentences and it found 92 grammatically incorrect out of 516.

As you can see above, the predictions on the grammatically incorrect sentences look pretty accurate.

Converting web pages to sentences to predict their grammatical correctness

Splitting text into sentences using regular expressions seems like a trivial thing to do, but there are many language nuances that make this approach impractical.

Fortunately, I found a fairly simple solution in this StackOverflow thread.

As you can see above, the technique works quite well. Now, we just need to feed these sentences to our grammar correctness predictive model.

Fortunately, it only found 4 out of 89 sentences grammatically incorrect in my last article.

Try this out in your own articles and let me know in Twitter how you do!

Opinions expressed in this article are those of the guest author and not necessarily Search Engine Land. Staff authors are listed here.

About The Author

Hamlet Batista is CEO and founder of RankSense, an agile SEO platform for online retailers and manufacturers. He holds U.S. patents on innovative SEO technologies, started doing SEO as a successful affiliate marketer back in 2002, and believes great SEO results should not take 6 months.
http://feeds.searchengineland.com/~r/searchengineland/~3/zDGpUXty1Fk/how-to-evaluate-content-quality-with-bert-337283