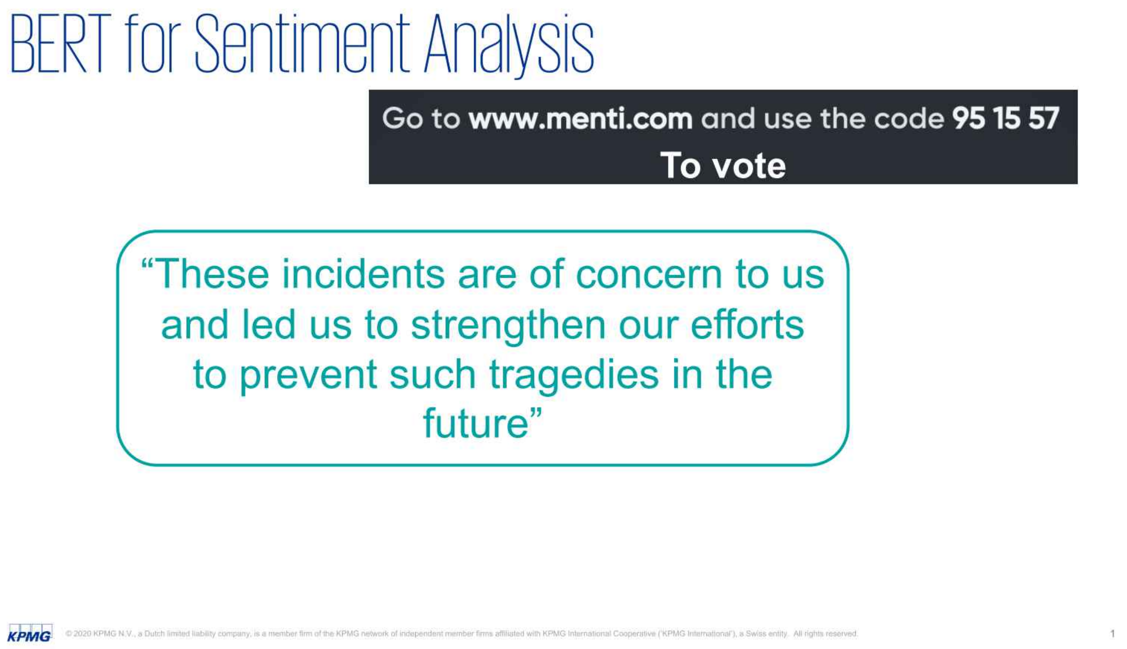
BERT for Sentiment Analysis on Sustainability Reporting



BERT for Sentiment Analysis

Go to **www.menti.com** and use the code **95 15 57**
To vote

“These incidents are of concern to us and led us to strengthen our efforts to prevent such tragedies in the future”

Software is changing the world. QCon empowers software development by facilitating the spread of knowledge and innovation in the developer community. A practitioner-driven conference, QCon is designed for technical team leads, architects, engineering directors, and project managers who influence innovation in their teams.

Transcript

Groothuis: Welcome to the data science track. I want to start off with a little vote. If you guys can get out your phones and vote on whether you think that this sentence is a positive, a neutral, or a negative statement? I'm getting a little feedback.

Sustainability Reports

We're going to talk about BERT for sentiment analysis. Not just normal sentiment analysis. No, we're going to look at sustainability reports. What exactly is a sustainability report? A sustainability report is basically a report that's in addition to the annual report, where companies publish about their economic, environmental, and social impacts, due to their everyday activities, and also about their vision and strategies surrounding these topics. We have a department within KPMG that is the sustainability department. One of the things that they do is they read these reports and they have basically an opinion about how good it is. They have to give a stamp of approval. The way that they do this is by a set of standards provided by the Global Reporting Initiative. They give these six metrics by which they judge whether or not a report is good enough. One of them is balance. A well-balanced report is when it reflects both the positive and the negative aspects of a company's performance, so that you can give a balanced overview to the stakeholders. It might not be in the company's best interest to talk about the negative aspects of the company. What tends to happen is that this balance is a little bit skewed towards the positive. Because what you often see is that statements, they like to pump up their positivity. You get statements like, "We create huge value for society." Or, "Our technological breakthrough will support the global battle against climate change." These are pretty strong statements. The job of my colleagues is to take these statements, go back to the colleague, and be like, "Maybe you should adjust this a little bit to reflect reality a little bit more".

The Problem

This is a tough thing. What they discovered is that they had some issues that they encountered while doing their regular job. One of them is that, different people have different opinions when it comes to sentiment analysis. For example, this might occur on a personal level. If you're in a bad mood, you might judge them more harshly than if you are in a better mood, or your workload is lower. The same thing happens with colleagues. Colleagues amongst each other might have different opinions. Of course, it also happens when you go into discussions with the client. They often will defend themselves and you have to tell them why you think they need to change. This is one of their problems. The second one is that this takes up a lot of time. They have to read the report, not once, but multiple times. These reports can be hundreds of pages. Finding these examples where they want to start a discussion takes a lot of time. The last thing, because of these two first issues, basically, it's very hard for them to do a comparison between different reports. To say that one report is well balanced but the next one isn't, is a really hard discussion. Also, you can't really compare to the same company's previous reports, or the year before that, to indicate changes or trends. This is stuff that you really want to do, but it's very hard to do at this moment. Basically, they asked us, can we quantify balance?

Sentiment Analysis

The first thing obviously that they thought about was sentiment analysis. They asked us, can you do sentiment analysis? We're like, "Of course we can." We did. Actually, we tried a lot. We found a whole array of different sentiment analysis models that are already out there, which are pre-trained. Some are commercial, like Heaven On Demand. Some are open source like the Stanford Sentiment Treebank. We basically tried all of them. Here is where I show the same report analyzed with all of these different models, with the sentiment aggregated per page throughout the entire document. This document has almost 60 pages. We saw cases where the model decided that the whole report was negative. We saw cases where they decided the whole report was positive. We saw continuous scale. We saw binary scale. What stood out the most is that we couldn't find any agreement within these models that it didn't seem to indicate a general pattern of positivity or negativity anywhere within the document, which is what we would expect if these models worked on our data. What was the problem? The problem is that most of these models are trained on different data that we were looking at. A very obvious dataset when you look into sentiment analysis is reviews. They're usually nicely written with strong voiced opinions. People either love a product or they hate it. That's when you tend to write a review. You automatically have your labels because they often come with a rating, or a star, or a score. These are typically the kinds of data that the model is trained on. This is not the data that we were looking at. We were looking at reports written by companies, which were closely related to annual reports. It's a very different language.

How to Define Sentiment

I showed this slide where I asked you guys if you wanted to score one of these things. I really like this result, because a lot of people voted positive. If you guys want to vote still, you can now. We can see there's already some disagreement. There's a few that are saying negative or neutral, but most actually are saying that this is a positive statement. We went to our sustainability colleagues, and we asked them, what do you define as a negative, neutral, or a positive sentiment that you want us to find? The positive is pretty obvious. Positive is bold statements. Anything that's overly positive, where they talk about their achievements, or some great value that they add to society. Neutral is factual information, anything that contains both positive and negative sentiments, like the stuff in between. A negative one was going to be the hardest to find because companies don't say our product is bad, or our service sucks. They say, we see a risk or we see a challenge. We see opportunities for improvement. The words used in these sentences could be viewed as positive. Most of you actually said that this previous sentence was positive, but by these definitions, this sentence is negative, because they're talking about a risk that they've identified, and they want to improve in the future. We needed a model that could handle this complex, natural language understandings. We needed something a little bit more sophisticated. We got BERT.

The BERT Model

What is BERT? BERT is a model which was trained and published by Google. It stands for Bidirectional Encoder Representations from Transformers. Of course, this is probably a backronym but that doesn't matter. What you can view BERT as is like a general language understanding model. You can use this model to do various NLP tasks.

Vector Representations

To understand how BERT works, I'm going to talk about vector representations. The way that you want to train a model is you do calculations. To do calculations, you don't need words, you need numbers. You need a way to represent those words into numbers. The very basic way of doing this is to create what's called a one hot encoding. You basically take a very long list of all of the words that appear in your text at least once. You give each word an index. Then you give each word a vector where you only put a 1 in the place of its index. Rome in this example, only has a 1 in the first index. Paris only has a 1 in the second index. Now we have a vector representation. Of course, these vectors don't contain any meaning, the way that we would understand the relationship between words. In this example, Rome and Paris are both cities. It's just coincidence. If I change the order of my list, I get different vectors. How do you solve this?

Word2vec

One of the first things that was done to build good vector representations for words was what's called a Word2vec model. The way that a Word2vec model works, is basically a variation of an autoencoder. An autoencoder has a neural network, which has the structure of an hourglass. You have a large input, a small stuff in the middle, and a large output. You try to predict the input itself. What happens as you get better is that this smaller stuff in the middle will create a smaller vector that can be used as a representation of whatever you're trying to predict, in this case, words. What a Word2vec model does is it loops over all of your text like a sliding window. It tries to predict the word that is in the middle using the words that are around it. Intuitively, you can see that this will create vectors that represent the words by their context. Words that are similar like king and queen will be close together, and this will be called a vector space or the latent space. The same way, we can see the relationship between king and queen, we can also see this relationship between man and woman. The same goes for verbs, so walking and walked will be close together. We now have representations for our words.

Sequence Models

There's one more problem. We have words which have different meanings, but are the same word. Let's take the word bank. If I say I walked down to the river bank, I'm talking about a geological location. When I say I walked out to the bank to make a deposit, I'm talking about a financial institution. The problem is that with this method, the vector representation of the word bank will not appear either with the geological locations or either with the financial institutions. It will appear somewhere in the middle. You lose some of the specific context that you're looking for. One of the ways that this was solved is by way of a sequence model. What this tries to do is given a sentence, you predict the next word. The word, word, will now have a vector representation that is actually dependent on the words that came before it. Now we have a good representation for the word, bank, if I'm looking at the sentence, I walked down to the river bank. However, if I look at the example, I walked out to the bank to make a deposit, the information containing that this is about a financial institution is now behind the word. We're only halfway there. We can simply solve this by also doing the same thing but backwards. We try to predict the word that came before it, and concatenate the results. This is good. This works really well for a while.

Masked Language Modeling (MLM)

Then came BERT. What actually is BERT doing that's a little bit different than this? What we now have is what's called a contextualized word embedding. BERT tries to create the same thing but in a little bit of a different way. BERT tries to predict words in the middle of a sentence by simply masking them. That way, you get the entire context of the sentence when trying to predict the word, which will then have a better representation using the entire context, and not just going back and forth, but the whole thing.

Next Sentence Prediction (NSP)

Not only that, BERT does a little bit more. It also is trained on what's called a next sentence prediction. What we see here is that we have two sentences. They are also surrounded by two specialized tokens. The first thing, the CLS token, and the second one, the SEP token indicate different things within this model. The SEP token is just to separate the first sentence from the next so the model knows where the next sentence begins. The first one, the CLS token is going to be very important because it's the classifier token. In the architecture of this model, the CLS token has its output, its own vector, which is used as input to predict whether or not this next sentence follows the first. They do this, obviously, on a whole bunch of different texts. Eventually, when you do this enough times, the CLS token will not just be a random representation, it will actually start to represent the entire first sentence. That's where we can use it to do classification.

Uses of the BERT Model

The BERT model can be used for different things. You can do something that's called named entity recognition, or part of speech tagging, where you want to recognize what words they are. You would do this on the output of the different tokens related to those words. Where we're looking at is classification. For classification, we want to have the representation of the entire sentence. We look at the output of the CLS token. What this would look like in training, is we have an input sentence. This is then tokenized, which you can see in the second one. We add the CLS token and the SEP token. The second sentence, in this case, for us doesn't exist. We don't need it. It's just left empty. We run the whole thing through the BERT model, which is pre-trained. Has all these embeddings. We use the output from that model into our classifying layer. This is then what we actually train our own labels, in this case, a positive one.

BERT Model Types

When BERT was first released, there were two basic models for English that they made. One is BERT-Base. It's a smaller model. It has 12 layers. Then the BERT-Large model, which has more layers. I have some beef with Google because I don't like the name BERT-Large, because I think it should have been called Big BERT, because it's Big BERT. I shouldn't complain too much, because over the past year or so, there's been a lot of new development on this topic. A lot of new models have come out that would use the BERT architecture but are trained on different languages, or they're optimized. For example, for French, we now have CamemBERT. There is also one for Dutch which is called RoBERT. There's a lite version of BERT called ALBERT, a light BERT. We also have another optimized version, tinyBERT. There's a whole bunch of other variations. There's so much progress still going on. Also, there have been plenty of implementations now into different libraries. There's an implementation in PyTorch, in Keras. Hugging Face is a really good place to find all of these embeddings ready for you. If you want to start out with using BERT, I would highly recommend checking out the Hugging Face repository.

Fine-tuning BERT

We're going to fine-tune BERT to our specific data. What is the data? The data that we had was almost 800 sustainability and integrated reports. Integrated reports just being an annual report, where the first half is about the sustainability topics. Most of them had around 90 pages. We extracted all the data, and the only pre-processing you have to do for BERT is split it up into sentences, which is quite nice. There's a lot of NLP pre-processing tasks that you could do, like stemming or limitization. It's all not necessary for BERT, you can just put in the entire sentence and there's a specialized tokenizer that will make sure that the model is able to handle whatever you give it.

TensorFlow

The original model by Google doesn't only come with the architecture and the weights of the model. It also comes with some scripts for you to run for a classifier or any other task that you want to do, except that the whole thing is written in TensorFlow. Personally, I don't really like TensorFlow. It can be very dense. Especially, this script that they've provided was a little bit hard to use for the simple use case that we had.

Model Implementation in Keras

The first step that we did is we implemented this thing in Keras, which is now already done for you, so you don't have to do it again. Keras has a much nicer interface for you where you can build up your model. You can do predict. You can do train. It's just a little bit more intuitive. Also, this was before TensorFlow 2.0, which now already has this layer of Keras in front of it. One of the nice things about Keras when you've built your model is that you can do this nice summary. You can actually see what the model looks like. This is a snippet of it. We can see the actual input and the output of our model. The input in this case is 50 by 768, where the 50 is referring to the number of tokens that we're putting into the model. Basically, the number of words. Then the 768 is the length of each of the vectors. It's quite a large vector space that has been created by the BERT model. Our output is only 3, because we're only interested in three labels. It will be just the probabilities of the sentences belonging to each of those labels.

Then we needed data, obviously. We went back to our sustainability colleagues, and we're like, "You want to create some labels for us?"

They spent two days going through a whole bunch of random sentences and gave them the labels that they wanted. We ended up with 8,000 labels. The negative sentiment was represented in the dataset. We were a little bit worried whether or not this was going to work. We crossed our fingers, threw the data into the model, and asked ourselves, "Is this going to work?" Yes, it worked.

Results

What were actual results? We ended up with an accuracy of 82%, which was decent. When we looked into the results for each of the labels, we saw that for negative we were doing 71%, positive 80%. Neutral, obviously, being there the most, was the highest. Where we were actually the happiest about is that the model didn't make any confusions between positive and negative. All of the 18% that it got wrong was always between a negative and a neutral statement, or a neutral and a positive statement. Since we already discussed that labeling the sentences is quite hard and there's a lot of gray areas, we were quite happy with the fact that the model was able to separate them at least in those two directions. Also, when we dug a little bit further into what it was actually getting confused about, we could see these two examples. The first one is something that BERT predicted as neutral, and was originally labeled as negative. When we showed this to our colleagues as well, they were like, it's fine. I'm not really mad about this one. The same was for the other one. They actually said it was a neutral statement, but BERT said it was a positive one, talking about an achievement we can assume. What we were most interested in is, was it able to detect this hidden negative sentiment? It was. Unlike you guys, it was actually able to predict correctly that this sentence was a negative statement.

The Problem

We created this model for them, and basically, we fixed the second one. We were now able to quickly go through all of these documents and present it for them. Give them some examples of problems or sentences they should pay attention to. What we didn't exactly solve is that different people have different opinions. We worked closely together with only two people from the sustainability department. Basically, it was their input that was put into the model. We needed to make sure that the model was generalized enough, and wasn't just copying these two people. What did we do? We asked other colleagues from the sustainability department to also label a bunch of sentences. We all gave them the same set. Then we checked how often they agreed. It was only 73% of the time that they actually agreed. Most interestingly, they even had cases where they didn't agree about the negative and the positive statements. They were reversing things. The next step, obviously, is checking how well is BERT holding up to this? It was actually doing better. We got 81% agreement with BERT and all of the people. What did this provide for them? It didn't solve their problem of different people having different opinions, but what it did give them is a stable way of now analyzing all of these reports, because BERT at least is going to agree with itself. You could do a bulk analysis of all of these things, so we have something to compare to, which was very important for them. Obviously, nowadays you can analyze one report. You can analyze a whole bunch of reports. You can analyze reports from the past, and reports that are coming in. We can make comparison with peers, competitors, or analyze trends.

Demo

This is just a demo. This is not what the actual interface for them looks like. It's just to give you an idea. We've named our thing, the subject extraction and sentiment analysis module, which is why it's called SESAME. What we can do is just upload a report. The first thing it does is just gives them a little preview of what the actual PDF looks like. Then the first step that we do, obviously, is extracting the raw text. This is just also to give them an overview of if everything is going right. We're usually analyzing reports that are made in PDF, so they're actually extractable. What sometimes happens is that these things are scanned, and then PDF creator already OCR's this data. Then it's a little bit iffy. It's just for them a visual check to make sure that everything's going correctly. This is supposed to be difference, but there's a little question mark there. This might pop up later. It's also to give them a visual. We've also added some other nice features, which is part of speech tagging, where you can actually see if something is a noun or a verb. The same thing with named entities. Not a named entities one is working ok, but it appears not to really like Dutch names that much. It thinks Iain Hume is part institution and part person. It's just a person. Most importantly, we now have the sentiment as well so we can show them what are the negative and the positive sentences that are labeled in this particular report. They can check it, see if they agree. If there's something that's weird about it. Also, it gives him these examples that he wants to discuss with a client. Furthermore, which is what they probably use the most, is that it gives them a nice overview. This is what we saw in the beginning is an aggregated view per page of the sentiment throughout the document. On top of that, we also have the topics here, which we've created with an unsupervised way of finding topics in documents. Additionally, if they do want to check here, "This is a weird spike. I want to know what happens here." Let's go to page 107. Then below it we can see all of the sentences and they can see which ones are labeled negative, and gives them an indication why and what it is about.

Topic Modeling

This is what it looks like for them. They like to work in Tableau. They've made their own dashboard with their own colors. Most importantly, they also are attached to the actual database so they can make these comparisons between peers, competitors. Here you see a comparison between KPMG, EY, Deloitte, and PwC. Apparently, we're a little bit more positive. I don't know what it tells you, but we are. Then we can also see that the sustainability reports tend to be a little bit more positive overall than the annual reports, which actually makes sense. Because we see a lot of the times that if we have an integrated report, that the financial part of the report usually does contain some negative sentiment, but not a lot of positive sentiment, probably because they're reporting on incidences, and all that stuff, which is already flagged as negative. That's why the annual reports tend to score a little bit lower.

Questions and Answers

Participant 1: I'm trying to understand what is the 'aha' moment here. Because word for sentiment analysis is quite commonplace, at least from what I understand. I think the turning point was when you used the team actually to do the labeling for them. Is there anything else that I'm missing in terms of what makes this a hard problem to solve? Because we had done something similar for contract analysis, research reports, analysis on different companies. I'm trying to see, what is the thing that you did differently?

Groothuis: We didn't do anything differently per se. It was just that we found that the models we were using at the time weren't able to especially detect this hidden negative sentiment that they were interested in. Because we needed just something that was better able to build these representations that actually made sense in that context. BERT was one of the first models that came along, that we were like, this can actually work. Obviously, this was created over a year ago. Since then, there's been an explosion in these kinds of analyses. You're right. This is now very commonplace. Usually, those analyses are done on reviews, or Twitter data. It's a different language.

Participant 2: In terms of your evaluation metric, you're looking at accuracy. Since you're doing classification, I wonder if that is a good metric to look at. Are you also looking at the precision, the recall, F1, other evaluation criteria?

Groothuis: We could. The reason that we were looking at accuracy mostly in the first place is just, one, it was the easiest. We already knew that this was not going to be perfect. We weren't going to be able to optimize for any of these metrics, because there was always going to be this gray area that we're going to have to cover. We were actually more interested in being on this vertical line of these confusion matrices. That's where we mostly were looking at. Then spot checking, what was it doing and why was it doing it?

Participant 3: I'm looking at a problem at the moment where I'm doing real-time classification of speech. If somebody's on the phone, we can say that sounds like a personal injury claim, in which case, we'll start to ask certain questions. My assumption was I'll go down a very similar route of we already have the text classified as the sentences, other words that come for a personal injury claim. I have the data to train. Does this feel like it's still state of the art in terms of solving these problems or should I be looking at other approaches?

Groothuis: Definitely. As far as this model goes, I can't say it's unparalleled, because there's this new model that came out a few weeks ago by Microsoft. It has over 17 billion parameters, and there's no way you can run that fast on a small instance. I think it's definitely still worth to check this out. Also, definitely check out the optimization versions of BERT, so tinyBERT or ALBERT. I think that might be a very good place to start.

Participant 4: BERT is still a relatively maybe complex or advanced model. There's also simpler models. What I was wondering, did you also benchmark it against the simpler things or did you start with BERT immediately?

Groothuis: We started out doing a whole bunch of analysis, which were these.

Participant 4: What was the uplift of using BERT then? What's the benefit of using BERT compared to the other ones?

Groothuis: The benefit was that this didn't work. Most of these times it was either these things are just all negative or all positive, or it was just saying everything was neutral, which is obvious since it's the biggest class. Sometimes it would get good results on the positive sentiments, because those are very obvious big statements. It was especially the negative ones that were hidden as like positively framed that were very difficult to catch. We only had good results when we used the BERT model.

See more presentations with transcripts