A state of the art semi-supervised approach for topics classification

**Abstract:**

In this paper we’ll present the Gram-Schmidt model, a semi-supervised Deep Learning model. We’ll demonstrate its performance in the binary classification task in topics, showing its potential to outperform supervised fine-tuned models such as FinBERT.

**Introduction:**

Binary classification refers to those **classification tasks that have two class labels**. Examples include: Email spam detection (spam or not). Churn prediction (churn or not). Conversion prediction (buy or not).

Binary classification for topics using a semi-supervised approach

Unlabeled data, when used in conjunction with a small amount of labeled data, can produce considerable improvement in learning accuracy. The acquisition of labeled data for a learning problem often requires a skilled human agent (e.g. to transcribe an audio segment) or a physical experiment (e.g. determining the 3D structure of a protein or determining whether there is oil at a particular location). The cost associated with the labeling process thus may render large, fully labeled training sets infeasible, whereas acquisition of unlabeled data is relatively inexpensive. In such situations, semi-supervised learning can be of great practical value

In the field of information extraction and retrieval, binary classification is the process of classifying given document/account on the basis of predefined classes.

a matter dealt with in a text, discourse, or conversation; a subject.

**Problem definition:**

In this paper we’ll tackle on the task of labelizing text in environment or not topic.

The supervised approach consist on finding a labelised dataset (environment or not) to train a neural network on.

**The Dataset:**

The dataset was collected from hugging face’s datasets. The environmental data was took from the climateBERT database and the non-environmental database was found from single tweet

In total we recovered 5000 tweets of which 2500 are environmental, and 2500 non-environmental.

**The supervised approach:**

The classical way to deal with binary classification is the supervised approach.  A supervised learning algorithm analyzes the training data and produces an inferred function, which can be used for mapping new examples. An optimal scenario will allow for the algorithm to correctly determine the class labels for unseen instances. This requires the learning algorithm to generalize from the training data to unseen situations in a "reasonable" way (see [inductive bias](https://en.wikipedia.org/wiki/Inductive_bias)). This statistical quality of an algorithm is measured through the so-called [generalization error](https://en.wikipedia.org/wiki/Generalization_error).

*Before we begin, let me point you towards the [github repo](https://github.com/ThilinaRajapakse/BERT_binary_text_classification" \t "_blank) containing all the code used in this guide. All code in the repo is included in the guide here, and vice versa. Feel free to refer to it anytime, or clone the repo to follow along with the guide.*

Although a very powerful approach, this method requires large training dataset in order to yield quality results. “Talk about finetuning”Google’s [BERT](https://arxiv.org/pdf/1810.04805.pdf) was a paradigm shift in natural language modeling, in particular because of the introduction of the pre-training / fine-tuning paradigm: after pre-training in an unsupervised way on a massive amount of text data, the model can be rapidly fine-tuned on a specific downstream task…

**Cosine similarity:**

Text Similarity is one of the essential techniques of NLP which is being used to find the closeness between two chunks of text by it’s meaning or by surface. Computers require data to be converted into a numeric format to perform any machine learning task. In order to perform such tasks, various word embedding techniques are being used i.e., Bag of Words, TF-IDF, word2vec to encode the text data. This will allow you to perform NLP operations such as finding similarity between two sentences to extract semantically similar questions from FAQ corpus, searching similar documents from the database, recommending semantically similar news articles.

**Model:**

Overview:

Using the cosine similarity we’re going to classify a text as environmental or not.

How are we going to do that? The first thing we’ll need is a dictionary containing some of the topic’s keywords. We’ll generate the embeddings of that dictionary, then we’ll calculate the average of those embeddings, let’s call it v. v will represent the semantic of this topic. Now for a text t, we calculate its embedding e then we find the cosine similarity between v and e. If cos(v,e) is superior to a certain threshold then v and e have similar semantic meaning and therefore our text t is indeed in this topic.

Since our topic is the environment, I took the liberty of collecting the following dictionary on environment.

Now in order generate quality embeddings we’re going to use sentence BERT.

Then we’ll calculate the average of the dictionary’s embeddings.

This is the vector representing the topic v.

Now let’s train our model:

Let’s take a look at our training dataset. It contains 100 tweets labeled as 1 (environmental) or 0 (not).

We’ll start by generating the texts’ embeddings. Then we’ll calculate the cosine similarity of each text’s embedding with v.

We’re almost there, now we have the list of the cosine similarity. The bigger the cosine similarity, the bigger the probability that the text is environmental.

Now can we classify the texts?

Well we should find a threshold above which the cosine similarity indicates that the topic is environmental. The easiest way to do so is to simply try many thresholds going from 0.01 to 1.

We get the best accuracy of 85% for a threshold of 0.15

Now our model is trained.

Let’s put it to test with the following parameters: threshold=0.15 and topic\_dict = {}

**GS Model:**

In order to improve the performance of our model we could consider generalizing the mathematical tools that are applied.

So now after we calculate the dictionary’s words’ embedding we don’t just calculate the mean in order to find a vector that represents environment. Instead we consider E the vector space generated by the words’ embedding. Now for a certain text embedding v we calculate the cosine of the angle between the embedding and the plane E. To do so we find the orthogonal projection of v on the vector space E then we calculate the cos(v, p\_E(v)).

On the practical side, as a means to calculate the orthogonal projection of v on E we do the Gram-Schmidt process on the dictionary’s embeddings to generate an orthonormal basis for E.

We obtain the list of the cosine similarity between each text and his projection on E.

Just like the previous model we measure the accuracy of our model with different thresholds ranging from 0.05 to 1 in order to determine which tweets are environmental and which tweets are not.

We get the best accuracy of 95% for a threshold of 0.55

**Results:**

On the test set the