Multilabel Text Classification using Transformer Models

Submitted By:
Komal Mistry

Introduction

Dataset : Kaggle Toxic Comments Multilabel Text Classification Dataset

Collection of Wikipedia comments which have been labelled by human raters for toxic behavior into following categories:

- `toxic`
- `severe_toxic`
- `obscene`
- `threat`
- `insult`
- `identity_hate`

The task at hand is multilabel classification as a comment can belong to multiple/ all of the labels.

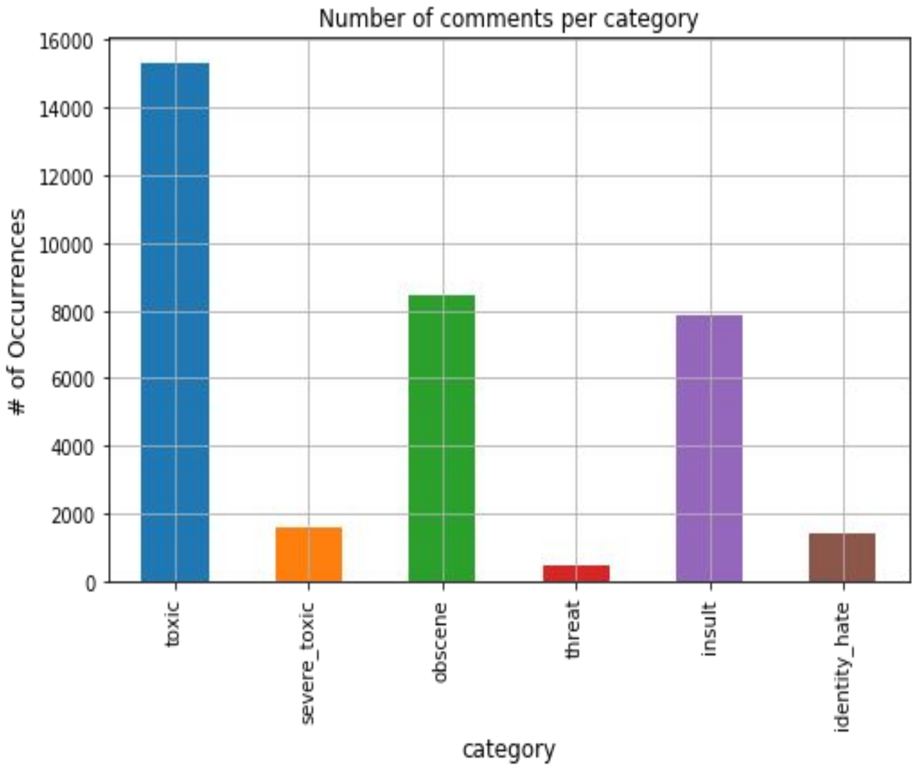
Project Breakdown

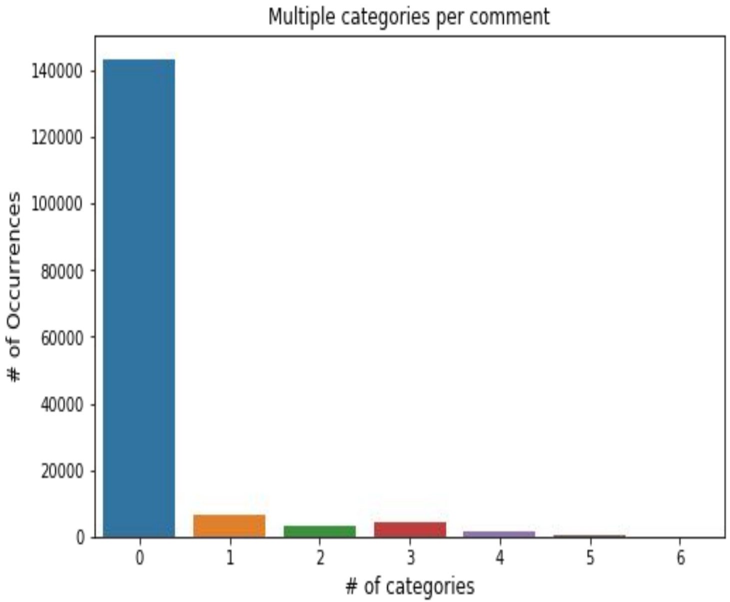
1. Data Exploration
2. Classification using sklearn models
3. Understanding word embeddings
4. What are Transformers?
5. Introduction to BERT, RoBERTa, XLNet, DistilBert
6. Simple Transformers
7. Tensorflow Keras BERT implementation
8. Other Evaluation Metrics

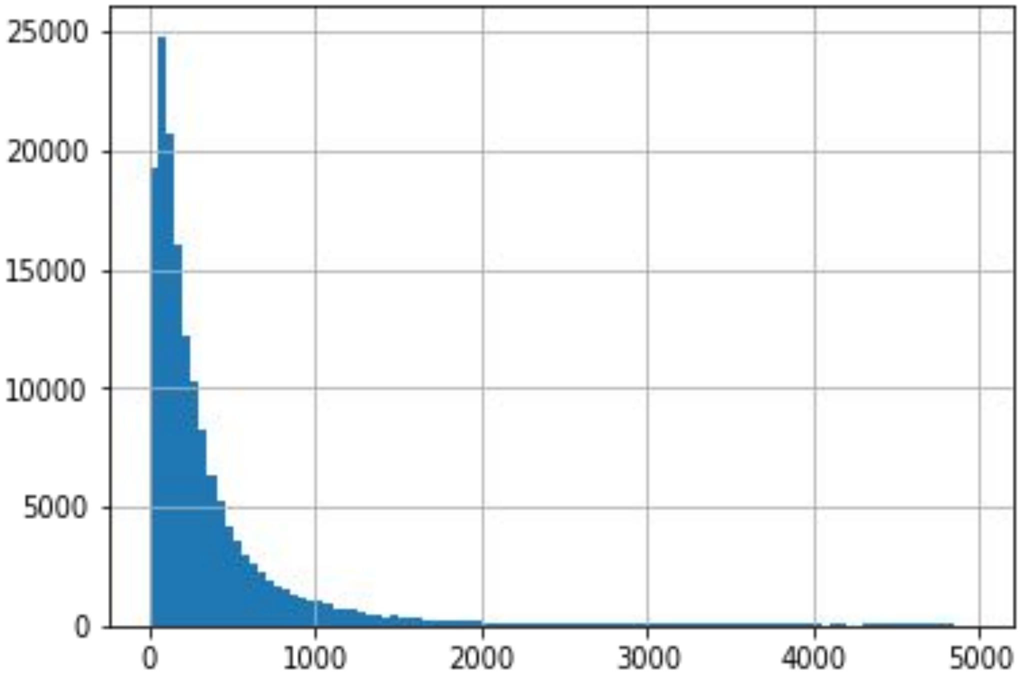
1. Data Exploration

During initial training data exploration we discover the following things:

- High Class Imbalance.
- 89% of the training data doesn’t belong to any label
- Most of the comments lengths are within 500 characters, some outliers with 5,000 characters.
- No missing data.


Number of comments per category
# of Occurrences
0
2000
4000
6000
8000
10000
12000
14000
16000
toxic
severe_toxic
obscene
threat
insult
identity_hate
category


Multiple categories per comment
# of Occurrences
0
20000
40000
60000
80000
100000
120000
140000
0
1
2
3
4
5
6
# of categories



25000
20000
15000
10000
5000
0
0
1000
2000
3000
4000
5000

2. Classification using sklearn models

1. Comments Preprocessing
2. Train-Test Split
3. Used TF-IDF vectorizer to convert all text into numbers
4. Integrated vectorizer and OneVsRest Classifier for Naive Bayes, LinearSVC, Logistic Regression.

2. Evaluation of Sklearn models

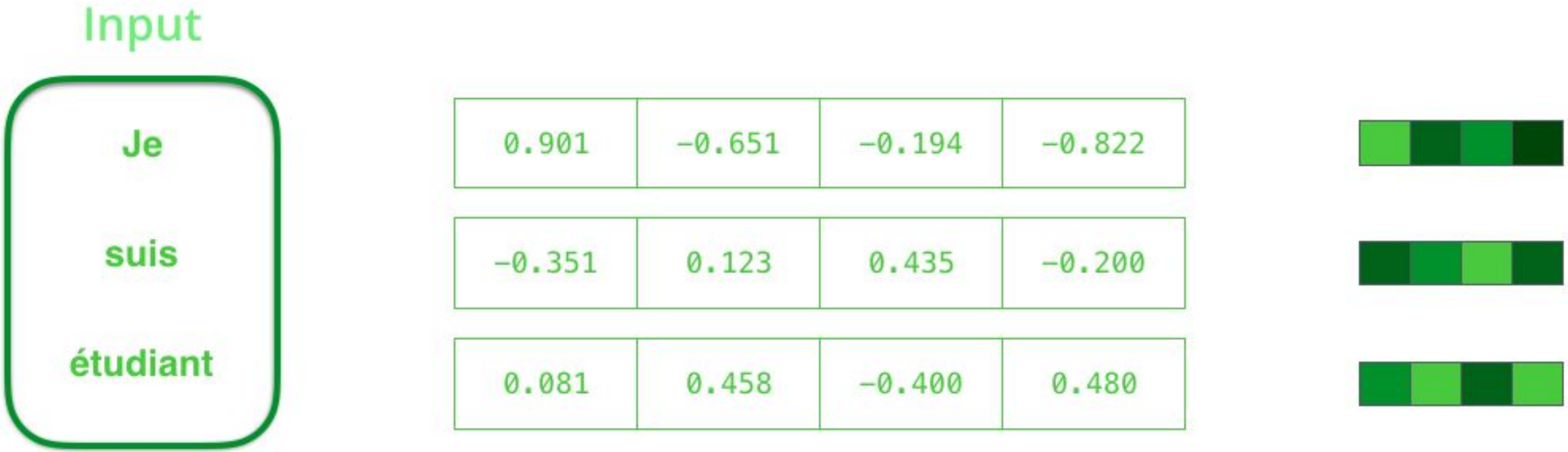
Evaluation Metric : Accuracy Score

Model	Toxic	Severe_Toxic	Obscene	Threat	Insult	Identity_Hate
Naive Bayes	0.919	0.990	0.951	0.997	0.951	0.991
Linear SVC	0.959	0.990	0.978	0.997	0.971	0.991
Logistic Regression Classifier	0.954	0.991	0.976	0.997	0.968	0.991

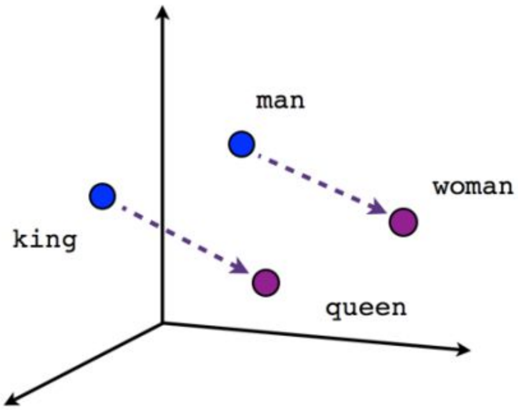
Why can’t we just use the sklearn models?

- Even though we are using OneVsRest classifier, internally it is still training a classifier for each class. Therefore to provide multilabel prediction for new comments, one would have to collate results from all 6 classifiers, making each prediction computationally expensive.
- Model will not generalize well to new data.
- There is no inherent semantic understanding of text in any of the classifiers.(We need word embeddings to do that)

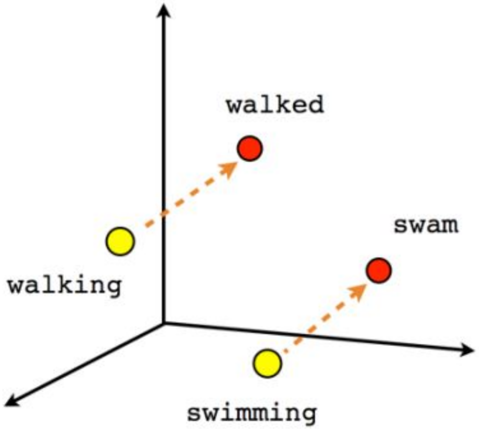
3. Word Embeddings



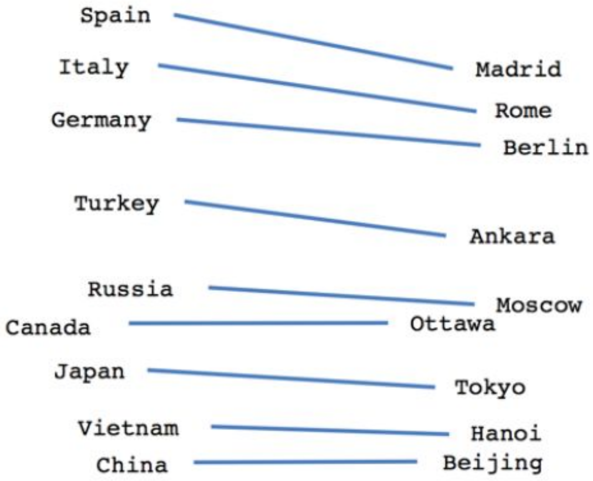
3. Word Embeddings



Male-Female



Verb tense



Country-Capital

Why are we not using pre-trained word embeddings?

We have the following word embeddings to choose from:

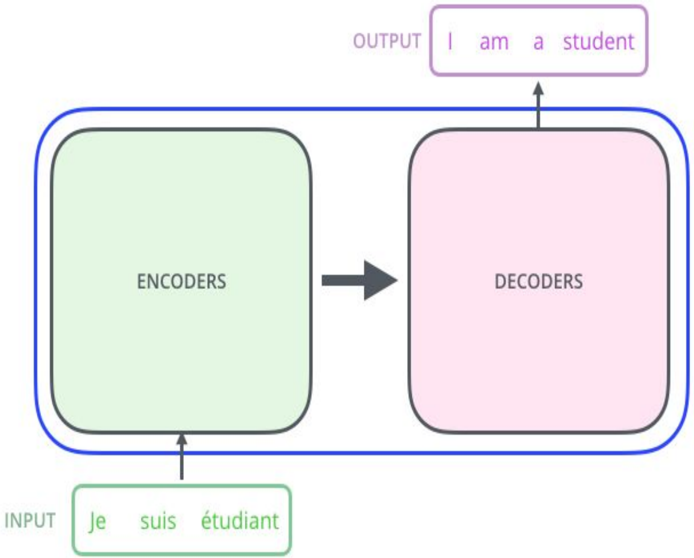
1. Word2Vec (Can access pre-trained embeddings from Gensim, but best performance only after training on dataset, local co-occurrence)
2. GloVe (creates global co-occurrence)
3. FastText (solves OOV problem by character ngrams)

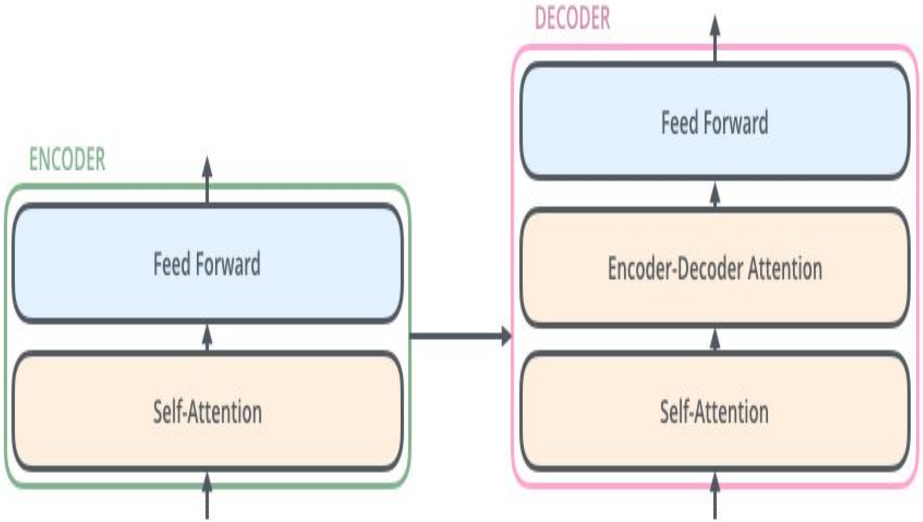
But Word2Vec, GloVe and FastText are context independent i.e they output just one vector for a given word combining all different senses of the word.

4. Introduction to transformers

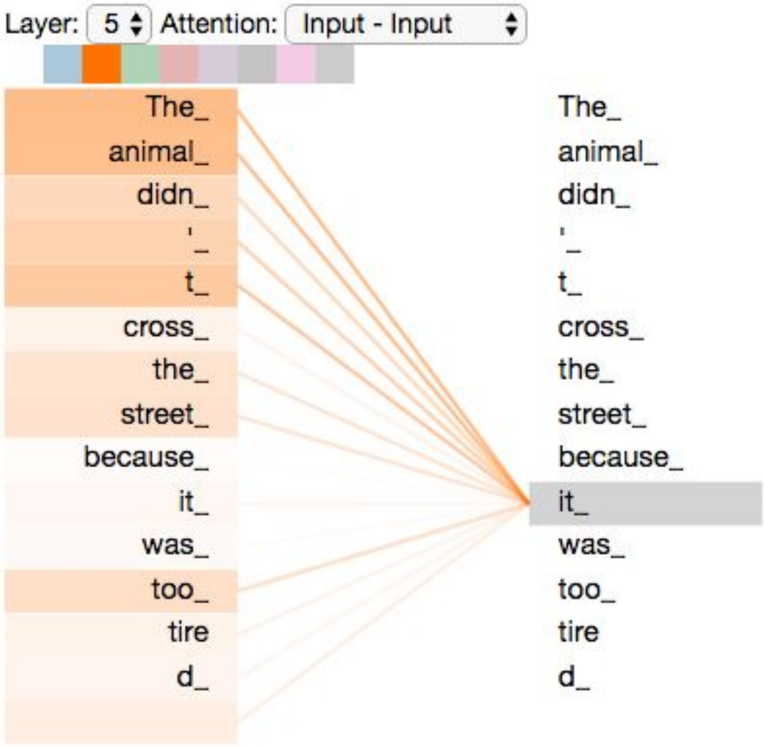
- Transformers are neural machine translation models.
- The building blocks of transformers are encoder and decoder.
- Let’s take a black box approach to understand how transformers are built.

Encoder - Decoder

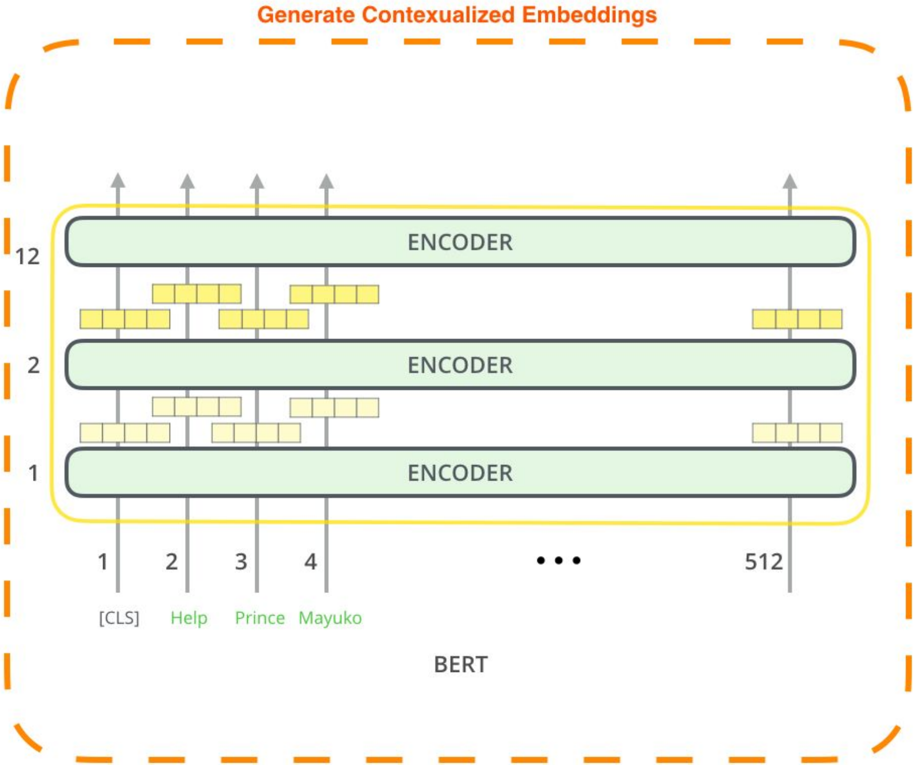




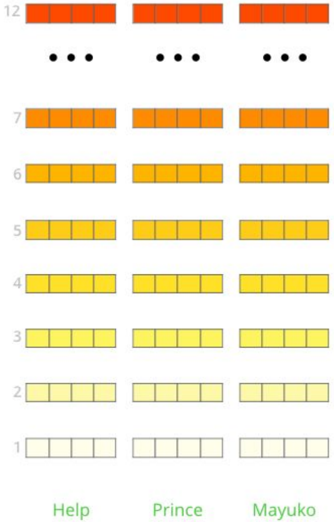
Attention Mechanism



BERT



The output of each encoder layer along each token's path can be used as a feature representing that token.



But which one should we use?

The magic of BERT

Apart from the large scale of the BERT models, there are other factors which contribute heavily to their superiority:

1. WordPiece Embeddings
2. Attention Mechanism (Creates better embeddings to capture semantics of the words)
3. Positional Encodings (Enables model to get better understanding about the meaning of a word based on it’s position in the sentence)
4. Scale and origin of the training corpus

5. Difference between BERT, RoBERTa, XLNet, DistilBERT

	BERT	RoBERTa	DistilBERT	XLNet
Size (millions)	**Base**: 110 **Large**: 340	**Base**: 110 **Large**: 340	**Base:** 66	**Base**: ~110 **Large**: ~340
Training Time	**Base**: 8 x V100 x 12 days* **Large:** 64 TPU Chips x 4 days (or 280 x V100 x 1 days*)	**Large:** 1024 x V100 x 1 day; 4-5 times more than BERT.	**Base**: 8 x V100 x 3.5 days; 4 times less than BERT.	**Large:** 512 TPU Chips x 2.5 days; 5 times more than BERT.
Performance	Outperforms state-of-the-art in Oct 2018	2-20% improvement over BERT	3% degradation from BERT	2-15% improvement over BERT
Data	16 GB BERT data (Books Corpus + Wikipedia). 3.3 Billion words.	160 GB (16 GB BERT data + 144 GB additional)	16 GB BERT data. 3.3 Billion words.	**Base**: 16 GB BERT data **Large**: 113 GB (16 GB BERT data + 97 GB additional). 33 Billion words.
Method	BERT (Bidirectional Transformer with MLM and NSP)	BERT without NSP**	BERT Distillation	Bidirectional Transformer with Permutation based modeling

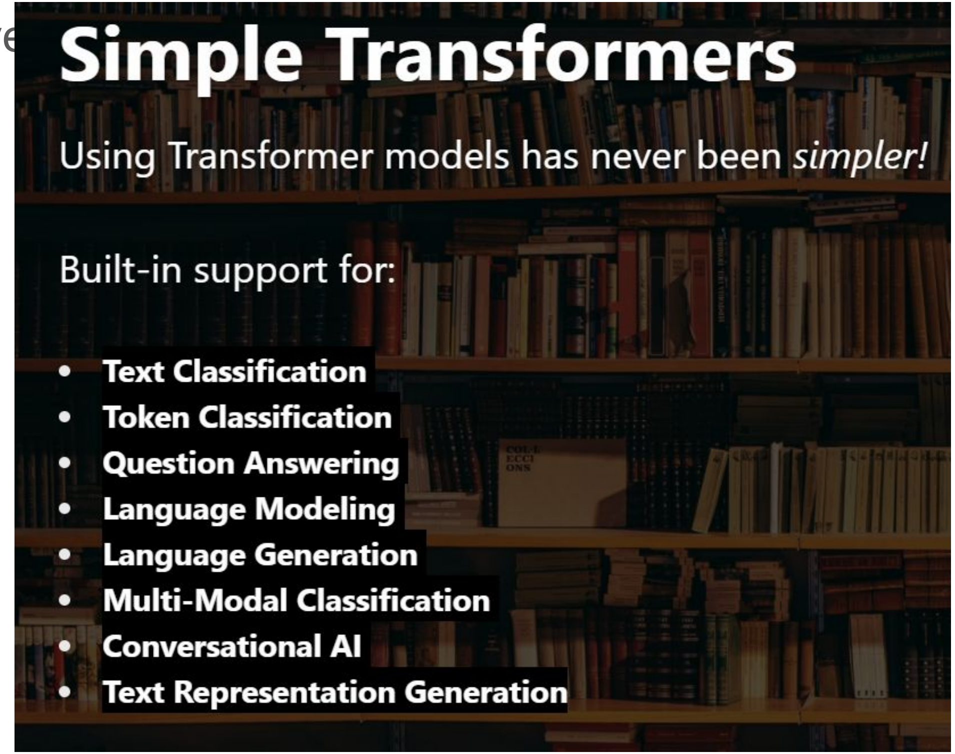
6. Introduction to Simple Transformers

- Developed by Thilina Rajapakse, ve
  early stage (267 users)

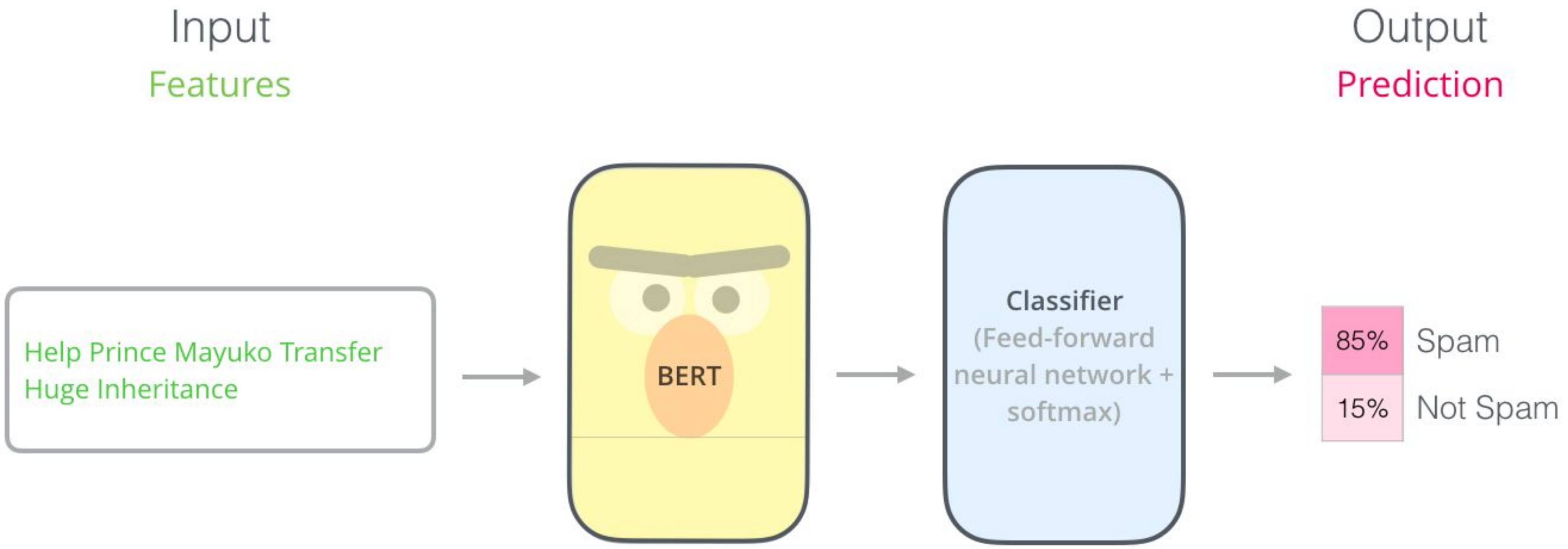
- Abstraction over HuggingFace transformers library.

- Runs on a Pytorch backend.

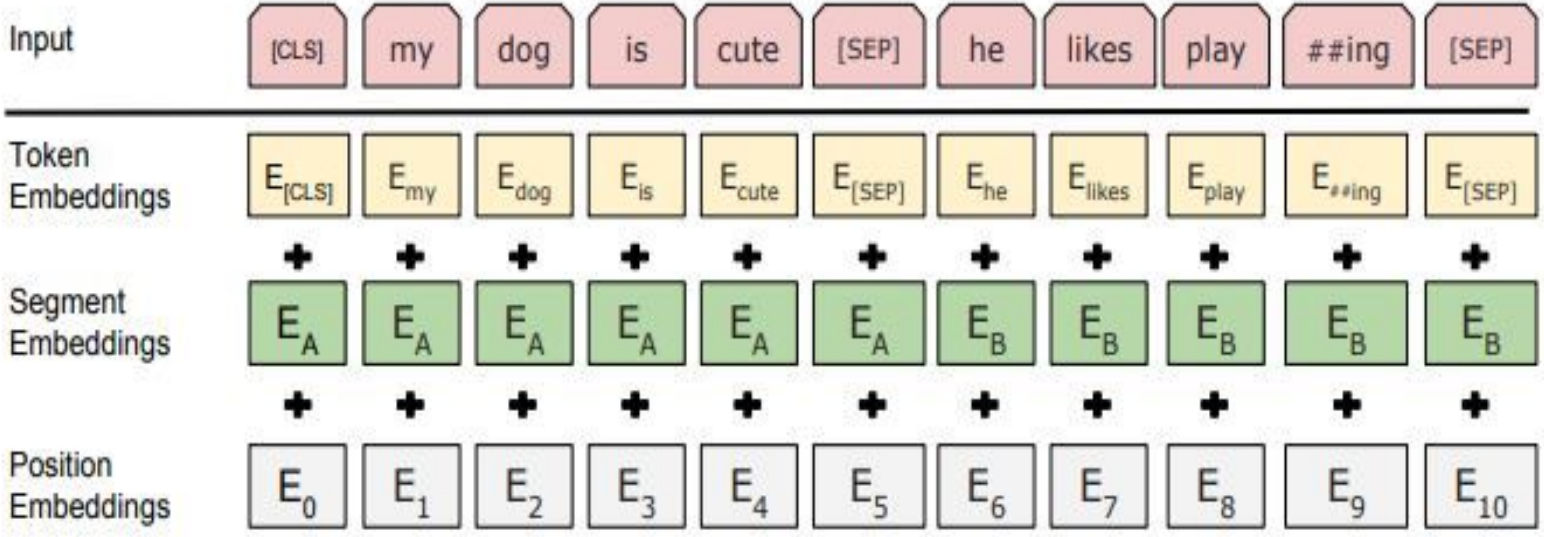
-



Architecture of Multilabel Classifier



Understanding Input to Transformer Models



Evaluation of Transformer Models

Evaluation Metrics : Label Ranking Average Precision(LRAP)

Model	LRAP (between 0 and 1)
Roberta Base	0.9978
Bert base uncased	0.9973
XLNet base	0.9957
Distilbert base	0.9949

Training Specs: trained on random subset of 1000 from train data for 3 epochs.

7. Tensorflow Keras BERT base

- The SimpleTransformers package gives us easy access to pre-trained models for multilabel classification but leaves very little room for modification.
- The Keras implementation has to include a lot of boilerplate code as compared to simple transformers but it allows one to define their own layers on top of the transformer models.
- We can add additional dropout layers before dense classification layer to avoid overfitting.
- The only challenge is to format the input data and create tensors for token embeddings and mask embeddings.

8. Other Evaluation Metrics

1. LogLoss
2. AUC
3. Microaveraging precision and recall
4. Hamming Loss

But I eventually decided to go with LRAP since it was recommended by the simpletransfromers documentation and it is more practical from label ranking perspective instead of just hard classification.

Next Steps

To further improve the classifiers, we have the following options:

- We have only used base transformer models, we can try using the full models by making hardware adjustments for memory and GPU usage.
- The huggingface library also contains transformer models that have been trained on several other text corpuses, one such model is : DistilRoberta_fine_tuned_tweets_hatespeech.
- This model is a distilroberta model trained on twitter hatespeech therefore it might be better at classifying hateful comments since it would have better embeddings and vocabulary for our specific genre of comments.

Index for trained models included in the project

Tensorflow bert base model trained on entire training dataset (3 epochs)

- Tensorflow_bert.h5 (t)

PyTorch model trained on train data subset (1000 comments, 3 epochs)

- Bert_pytorch_model.bin
- Roberta_pytorch_model.bin
- Xlnet_pytorch_model.bin
- Distilbert_pytorch_model.bin

References

- https://jalammar.github.io/illustrated-transformer/
- https://pysnacks.com/machine-learning/bert-text-classification-with-fine-tuning/#what-is-bert
- https://simpletransformers.ai/
- https://towardsdatascience.com/bert-roberta-distilbert-xlnet-which-one-to-use-3d5ab82ba5f8
- https://jalammar.github.io/illustrated-transformer/
- https://huggingface.co/exbert/?model=bert-base-cased&modelKind=bidirectional&sentence=The%20girl%20ran%20to%20a%20local%20pub%20to%20escape%20the%20din%20of%20her%20city.&layer=11&heads=..0,1,2,3,4,5,6,7,8,9,10,11&threshold=0.7&tokenInd=null&tokenSide=null&maskInds=..&hideClsSep=true
- https://github.com/susanli2016/Machine-Learning-with-Python/blob/master/Multi%20label%20text%20classification.ipynb
-