Trainable Information Extraction.

Wenjie Jin - 219516
Rodrigo Caero - 182514
Roger Pedrós - 206798

Universitat Pompeu Fabra
11 December, 2021.

Introduction.

As a part of the subject Intelligent Applications for the Web taught by Horacio Saggion, we have been learning about the field of information extraction. During the course, we have seen different ways of extracting and understanding information, up to advanced solutions based on BERT.

During the course, we reviewed by groups different papers related to the use of BERT. That is why as part of this work we have been working on a BERT model which we had to adjust to finally incorporate into an application.

For our model we work with a pre-trained BERT model, then to identify certain aspects of a text we implement a fine-tuning process using the *Conscius Corpus* dataset. Specifically, in our case, we work with an aviation accidents data set.

For our application, we have built a CLI system that extracts information from a text, as specified by our teacher. Additionally, we have built a small web application, with our model, running on a Flask server in order to test it more comfortably.

Description of the dataset used.

We work with the Concisus Corpus dataset (Saggion & Szasz, 2012) given in class. As mentioned above, we work only part of the aircraft accidents. El dataset has a sample of 31 documents and has already annotated the information that we need to extract.

The database includes up to 25 tags, that can be resumed as:

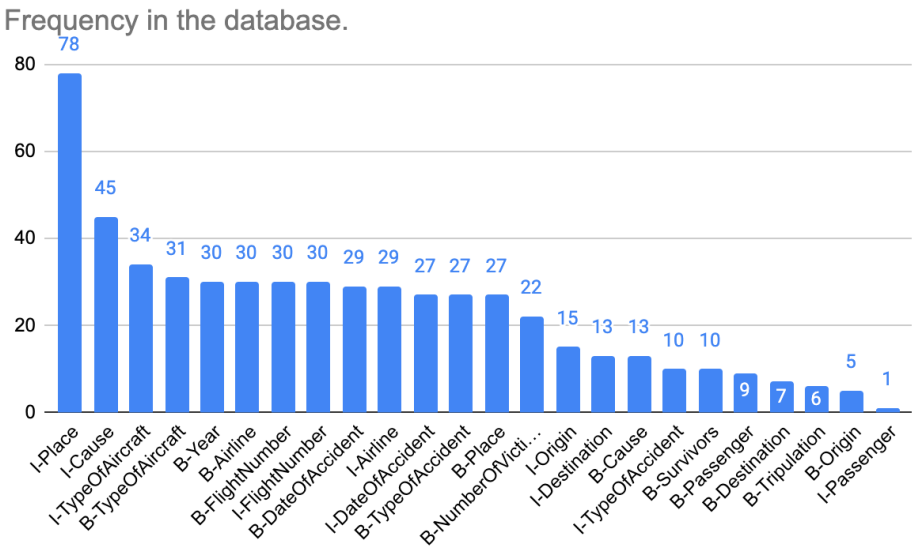
DateOfAccident, Year, TypeOfAccident, NumberOfVictims, Survivors, Passengers, TypeOfAircraft, Tripulation, Place, Cause, Airline, FlightNumber, Origin, Destination.

But the laboratory documents emphasize the specific use of these:

Airline; DateOfAccident; FlightNumber; NumberOfVictims; Place; TypeOfAccident; TypeOfAircraft; Year.

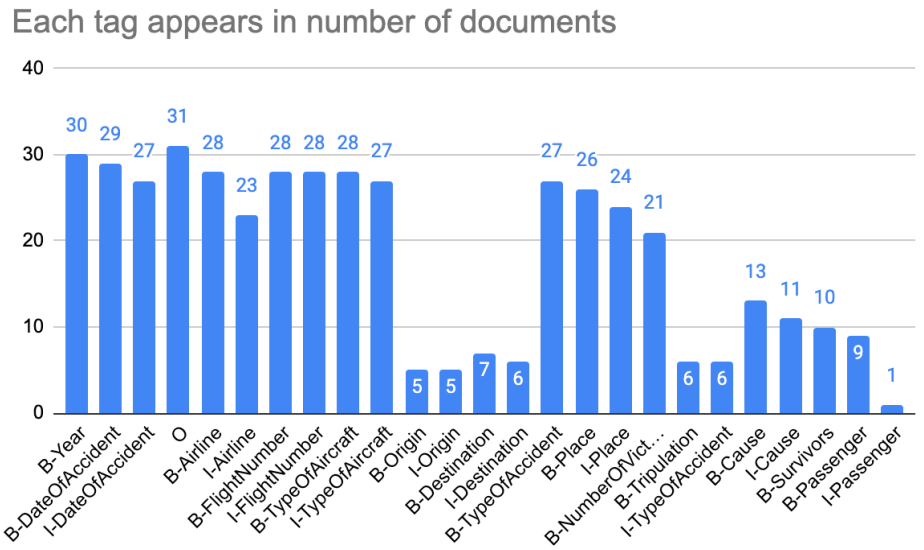
The frequency of the tags:

Frequency	
O	729
I-Place	78
I-Cause	45
I-TypeOfAircraft	34
B-TypeOfAircraft	31
B-Year	30
B-Airline	30
B-FlightNumber	30
I-FlightNumber	30
B-DateOfAccident	29
I-Airline	29
I-DateOfAccident	27
B-TypeOfAccident	27
B-Place	27
B-NumberOfVictims	22
I-Origin	15
I-Destination	13
B-Cause	13
I-TypeOfAccident	10
B-Survivors	10
B-Passenger	9
B-Destination	7
B-Tripulation	6
B-Origin	5
I-Passenger	1



The O tag has been excluded from the chart since it's the most abundant but the less valuable tag.

Each tag appears in number of documents	
B-Year	30
B-DateOfAccident	29
I-DateOfAccident	27
O	31
B-Airline	28
I-Airline	23
B-FlightNumber	28
I-FlightNumber	28
B-TypeOfAircraft	28
I-TypeOfAircraft	27
B-Origin	5
I-Origin	5
B-Destination	7
I-Destination	6
B-TypeOfAccident	27
B-Place	26
I-Place	24
B-NumberOfVictims	21
B-Tripulation	6
I-TypeOfAccident	6
B-Cause	13
I-Cause	11
B-Survivors	10
B-Passenger	9
I-Passenger	1

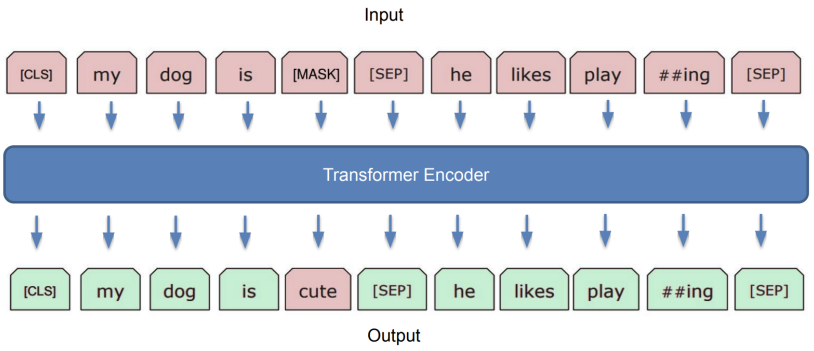


Description of the software.

The software that we used to develop our fine-tuned model is basically BERT over the Pytorch framework, running over Google Colab. We have also used a set of basic Python libraries to facilitate development such as *pandas*, *numpy*, *sklearn*, *nltk*, *matplotlib*, *xml.etree*, and others. The data set that we used, as we mentioned, is the CONCISUS Corpus.

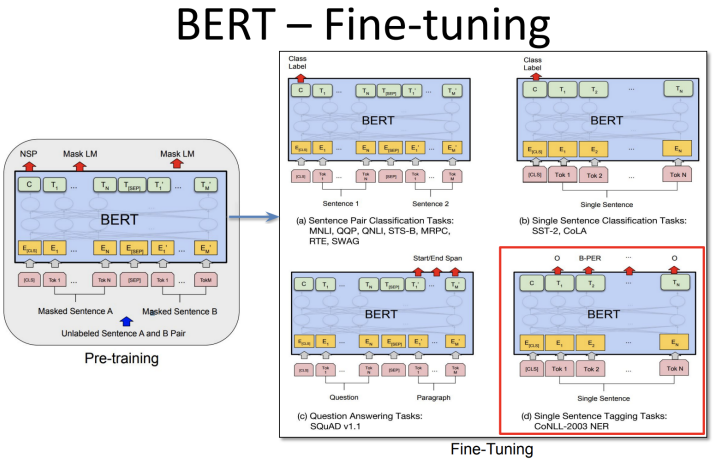
BERT: Bidirectional Encoder Representations from Transformers is a transformer-based machine learning technique for natural language processing (NLP) pre-training developed by Google. BERT was created and published in 2018 by Jacob Devlin and his colleagues from Google. (Devlin, 2018). The main technical innovation of BERT is the application of the bidirectional training of Transformer, a popular model of attention, to the modeling of the language. Contrasting the previous efforts, which were usually based on rule-based systems or linked to human structures.

In our laboratory, we work with a pre-trained version of BERT specific for Token Classifications sourced by HuggIngFace. (HugginFace, 2020).



(Devlin, 2018).

The fine-tuning procedure allows adjusting a pre-trained model, using transfer learning, to specialize the network towards a specific content or format.
(Deeplizard, 2017).



PyTorch: Is an open-source machine learning framework used for AI and deep learning tasks, such as computer vision, natural language processing. PyTorch's tensors can be operated on a CUDA accelerated GPU.

Google Colab: Also known as Google Colaboratory, allows running and program in Python in your browser with different advantages: No configuration or installations are required, allow content sharing easily, gives free access to GPUs. Colab works similar to Jupiter notebooks, but all the computation power relies on Google Cloud, including GPUs and TPUs.

Colab can make Python executions easier, for students, data scientists, or AI researchers.

Implementation of the different programs

In this project, the first thing we do is to extract all the information we need from the XML file, and convert it to BIO format. And stored in a file (the “aviation.txt”) for later work. In our case, we have 31 sentences. We split it into 24 train sentences, 3 validation sentences, and 4 test sentences.

We used the pre-trained model BERT and made some fine-tuning to it. In fine-tuning, we set the maximum sequence length for our dataset to 128, because the maximum length of the sentence from our training set is 58, and in order to prevent users from inputting too long sentences, we added a certain amount of Padding. Finally, we generated three data sets (training, validation, and testing) respectively.

Furthermore, we set the number of samples selected in one training session (batch size) to 4. Set the learning rate 0.00001 to make sure we don't miss any local minima, Adam optimization algorithms, and training it 35 epochs. And after the model is trained and with a good result, we save it in “*model/aviation-accidents.pt”* format for easy use in our application.

In addition, in order to make our application more convenient to run, we also store the *max_seq_len* and the *index2tag* value in a file named “./model/“data.json”.

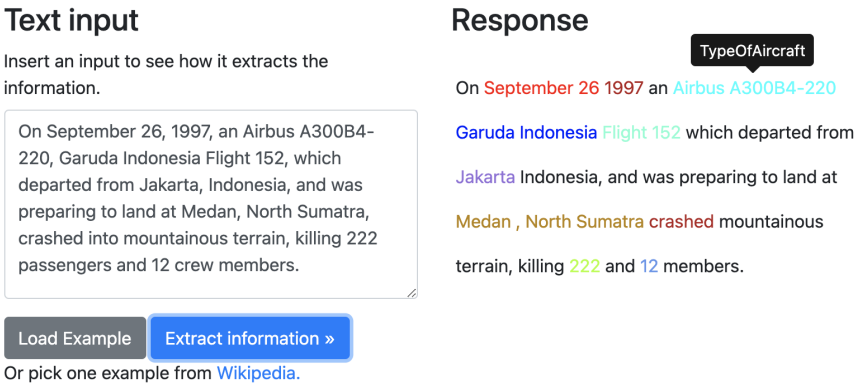
In the software, the user needs to give a new sentence, and then our software will return an extracted news summarizer. At the same time, we designed a method to extract text features that conform to the BIO format for the final result. And to allow our software to give more intuitive and easy-to-understand results. We implemented 2 different user interfaces to display the final result.

- **Application Python:** In this method, users only need to run the python application (the "Application.py "), introduce a new sentence and then the program will use the command-line interface to display the result. And introduce *QUIT* to exit the program.

```
READY FOR YOUR TEXT: On September 26, 1997, an Airbus A300B4-220, Garuda Indones
ia Flight 152, which departed from Jakarta, Indonesia, and was preparing to land
 at Medan, North Sumatra, crashed into mountainous terrain, killing 222 passenge
rs and 12 crew members.
CALLING THE EXTRACTION SYSTEM...

THE DATE OF THE AVIATION ACCIDENT WAS: September 26
THE ACCIDENT HAPPENED IN 1997
TYPE OF AIRCRAFT WAS: Airbus A300B4-220
THE AIRLINE WAS: Garuda Indonesia
THE FLIGHT NUMBER WAS: Flight 152
THE ACCIDENT TOOK PLACE IN: Medan , North Sumatra
TYPE OF ACCIDENT WAS: crashed
THERE WERE 222 VICTIMS
```

- **Web interface:** We wrote a simple web application. Using web framework Flask to build a localhost server and Javascript to communicate with the Python program, and using Bootstrap to design our web pages. In this case, users only need to enter the sentence and click the "*Extract information*" button. The system will transfer the content to the background python program through the *GET* method and dynamically update the page through Ajax technology.



Finally, about the final output. We decided to display all content that conforms to the BIO label format. Because it is an extractive summary system, we believe that without calculating relevance and importance, we need to show users all important information in the news.

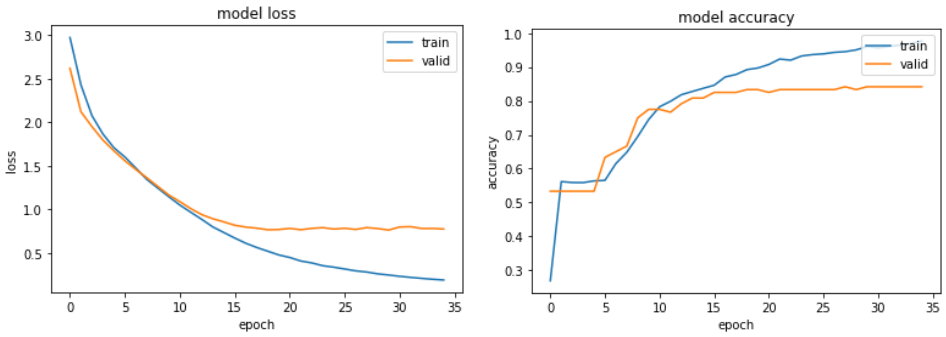
GitHub repository: https://github.com/Pedroos46/Aviation-accidents-IE.git which contains the Flask server with the web application, our model and the necessary data to run it, and the CLI application.

Conclusion.

We are broadly satisfied with the results of the experiment, we are employing a pre-trained model, which makes it possible with relatively little input information to find coherent and meaningful lexical relationships. So after training, the model got a very good result. The performance of the model when performing inference, and running it into production, is excellent, even on machines that do not have GPU acceleration.

Even though the input that needs to be processed is likewise fairly brief, and that may be a bit small, as it has a limit of 58 vocabulary pieces. On the other hand, while testing our model, we have detected inconsistencies in identifying words or small errors. We believe that this could be solved with a larger database, or where the texts are slightly different and diverse.

Also, as it can be seen in the picture below, we can see that after 35 training, the loss value of the train data set is close to 0 and the accuracy is close to 100%. Although this caused a certain amount of overfitting. But we think this is inevitable because our database is too small and the information that needs to be processed is too simple.



Even so, we still got an acceptable result with the validation data set. It can be seen that we have very high accuracy in most of the tags. Except for some tags that we do not need, for example: “Survivors” or “passenger”.

Validation : loss 0.87 accuracy 0.84

	precision	recall	f1-score	support
Airline	0.75	0.75	0.75	4
Cause	0.33	1.00	0.50	1
DateOfAccident	0.75	0.75	0.75	4
FlightNumber	0.80	1.00	0.89	4
NumberOfVictims	0.50	0.67	0.57	3
Passenger	0.50	0.50	0.50	2
Place	0.50	1.00	0.67	3
Survivors	0.00	0.00	0.00	1
Tripulation	1.00	1.00	1.00	1
TypeOfAccident	0.67	0.67	0.67	3
TypeOfAircraft	0.67	1.00	0.80	4
Year	0.75	1.00	0.86	3
micro avg	0.63	0.82	0.71	33
macro avg	0.60	0.78	0.66	33
weighted avg	0.65	0.82	0.71	33

In the future, we hope to solve this overfitting problem through deep learning technology. E.g, add a regularization term, generate multiple similar content, dropout, or find a new database with strong text features and more content. And calculate the word relevance so as not to output too much content.

Involvement of each of the members

In this laboratory, every member has participated in every step. Thanks to our continuous online meetings, we have been discussing all the changes.

But to break down the tasks by person:
The extraction data part has been mostly developed by: Wenji and Rodrigo.
The model and training part has been mostly by: Wenji and Roger.
The application part (terminal and web) has been mostly developed by: Roger and Wenji.
The report/questions and documentation has been mostly by: Rodrigo and Roger.

References.

- *BERT WordPiece Tokenizer Tutorial*. (n.d.).
  - Towards Data Science. Retrieved December 6, 2021, from https://towardsdatascience.com/how-to-build-a-wordpiece-tokenizer-for-BERT-f505d97dddbb

- Deeplizard. (2017, Nov 22).
  - *Fine-Tuning A Neural Network Explained*. Machine Learning & Deep Learning Fundamentals. Retrieved Des 2, 2021, from https://deeplizard.com/learn/video/5T-iXNNiwI

- Devlin, J. (2018, Oct 11).
  - *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. Arxiv.org. Retrieved Des 2, 2021, from https://arxiv.org/abs/1810.04805v2

- HugginFace. (2020). *BERT*.
  - BERT. Retrieved Des 2, 2021, from https://huggingface.co/transformers/model_doc/BERT.html#BERTfortokenclassification

- *Inside–outside–beginning (tagging)*. (n.d.).
  - Inside–outside–beginning (tagging). https://en.wikipedia.org/wiki/Inside–outside–beginning_(tagging)

- Jagtap, R. (n.d.).
  - *BERT: Pre-Training of Transformers for Language Understanding*. https://medium.com/swlh/BERT-pre-training-of-transformers-for-language-understanding-5214fba4a9af

- Saggion, H., & Szasz, S. (2012).
  - *The CONCISUS Corpus of Event Summaries*. ACL Anthology. Retrieved Des 2, 2021, from https://aclanthology.org/L12-1372/

- *Understanding BERT — Word Embeddings | by Dharti Dhami*. (n.d.).
  - Medium. Retrieved December 6, 2021, from https://medium.com/@dhartidhami/understanding-BERT-word-embeddings-7dc4d2ea54ca