Optional Project

Affect French Language model

Author:
Ayyoub EL AMRANI

Supervisor:
Dr. Pearl Pu Faltings
Yubo Xie

June 6, 2019

Contents

1 Introduction **3**

2 French Opensubtitles Data **4**
2.1 Overview of OpenSubtitles Data 4
2.2 Description and Parsing of Data 5
2.3 Analysis on Data 7

3 Sequence to sequence model **11**
3.1 Definition 11
3.2 Pre-processing and training 12
3.3 Testing the model 14

4 Comparison between DeepMoji and LIWC **15**

5 Conclusion **16**

Abstract

Human verbal communications and dialogues always hide several intentions and emotions.[1] Lots of research have been made in this field regarding the English language. However, the French language knows poor interests in this area. That's why, we were highly interested in implementing a french language model able to add emotions in generating sentences from a certain input of sentences. However, this ultimate goal was difficult to reach in the given time slot. Hence, in this report, we will be presenting the results we find regarding the work made from scratch. Indeed, we first had to look for french data on which the model can be based. We then proceed to a work of parsing and cleaning of the huge amounts of data found. We then did some basic statistical analysis on our data. We finally tried to test a "sequence to sequence" -hoping this can be integrated in a chatbot later- model on a small sample of our data to have an idea of the "quality" of our data. In parallel with this work, we did some research in order to test the validity of some potentially "good" emotion dictionaries. Indeed, we compared DeepMoji and LIWC dictionary and found out that LIWC dictionary was performing very poorly. This project is an "optional project" at EPFL during the spring semester 2019 supervised by Dr. Pearl PU Faltings.

1 Introduction

Movie and TV subtitles constitute a pioneer resource for the compilation of parallel corpora. Indeed, it is a resource full of information and rich of data since it represents a huge panel of conversations that is very comparable to our daily dialogues and conversations. From a linguistic perspective, subtitles cover a wide and interesting span of genres, from colloquial language or slang to narrative and expository discourse (as in e.g. documentaries). Large databases of subtitles are also available and continue to grow rapidly – for instance, the Open-Subtitles database contains more than 3 million subtitles in over 60 languages.[2]

The second section will focus on looking into the French Open-subtitles Data. it is a wide source of data that we will parse and analyze. To begin, French Open-subtitles Data is going to be parsed, cleaned and then analyzed.

In the third section right after, we will preprocess the data in order to train a one turn natural language processing model on it (a model that can predict one sentence given one sentence as input) -sequence to sequence in our case-. We will briefly explain the model, go through our preprocessing and present some results.

In the last section, we made some investigations/comparisons over existing models that provides emotions.

2 French Opensubtitles Data

2.1 Overview of OpenSubtitles Data

The OpenSubtitles data is a compressed cluster of folders containing XML files. Each XML file is split into a script portion with the subtitles of the movie and a metadata portion with additional information about the movie or show. The name of one of the parent folders of the XML file is the corresponding IMDb identifier of the movie or show, thus allowing us to extract additional information from the IMDb dataset.

The dataset consists of 66.5 GB (uncompressed) of XML files distributed in the following file structure:

```
|---- opensubtitle
|     |---- OpenSubtitles2018
|     |     |---- Year
|     |     |     |---- Id
|     |     |     |     |---- #######xml.gz
|     |     |     |     |---- #######xml.gz
|     |---- en.tar.gz
|     |---- fr.tar.gz
|     |---- zhcn.tar.gz
```

where:

- ‘‘ is a 6-digit unique identifier of the file on the OpenSubtitles dataset.
- Year‘ is the year the movie or episode was made.
- ‘Id‘ is a 5 to 7 digit identifier (if it’s 7-digit it’s also an IMDb identifier).

The subtitles are provided in different languages. In our work, we focus on the french data. It is important to note that the decompressed XML files vary in size, ranging from 5KB to 9000KB sized files.

Size	66.5 Gb
files	127204
tokens	791M
sentences	106.8M

Table 1: Basic stats on French Open-subtitles data

2.2 Description and Parsing of Data

As we said before each XML file is split into a ‘document‘ and ‘metadata‘ section. In our work, we don't give much importance to the ‘metadata‘ part so we will only focus on ‘document‘.

The ‘document‘ section contains all the subtitles and its general structure is the following:

```
|---- s
|     |---- time: Integer
|     |---- w: String
```

An example snippet of an XML file:

```xml
  <s id="1">
    <time id="T1S" value="00:00:51,819" />
    <w id="1.1">Travis</w>
    <w id="1.2">.</w>
    <time id="T1E" value="00:00:53,352" />
  </s>
```

The subtitles in each XML file are stored by blocks denoted by ‘s‘ with a unique ‘id‘ attribute (integers in increasing order starting at 1).

Each block (<s id="1"> for instance) has a:

- Set of timestamps (denoted by ‘time‘) with
  - A timestamp ‘id‘ attribute that can take two different formats: ‘T#S‘ or ‘T#E‘, where ˙S˙ indicates ˙start˙, ˙E˙ indicates ˙end˙ and ˙#˙ is an increasing integer.
  - A ‘value‘ attribute which has the format ‘HH:mm:ss,fff‘.
- Set of words (denoted by ‘w‘) with:
  - - an ‘id‘ attribute that is simply an increasing number of decimal numbers of the format ‘X.Y‘ where X is the string id and Y is the word id within the corresponding string
  - a non-empty ‘value‘ attribute that contains a token: a word or a punctuation character.
- It sometimes also has an ‘alternative‘, ‘initial‘ and ‘emphasis‘ attribute:
  - The ‘initial‘ attribute generally corresponds to slang words or mispronounced words because of an accent such as ‘lyin’‘ instead of ‘lying‘.
  - The ‘alternative‘ attribute is another way of displaying the subtitle for example ˙HOW˙ instead of ˙how˙.
  - The ‘emphasis‘ attribute is a boolean.

2.3 Analysis on Data

While Parsing the data and transforming it into several txt files -each file is a parsed movie file-. We Skipped a line every time a threshold on time difference between two sentences was verified. This, somehow enabled us to chunk the movie into "Dialogues". In order to define the threshold, we made a big analysis on all the corpus. We studied the difference of timing between each sentence on all movie files all together. Indeed, in order to complete this task, we followed the procedure below :

- Loop over 66.5 GB of French Open-subtitles data.
- Parse all files and extract the desired timings: $\Delta = sentence_i[time_{start}] - sentence_{i-1}[time_{end}]$
- Write the differences in a big txt file (1.6 GB)
- Perform the Analysis.
- Get the histogram of counts over the differences of time.

The following histogram shows us the count of these different Δ:

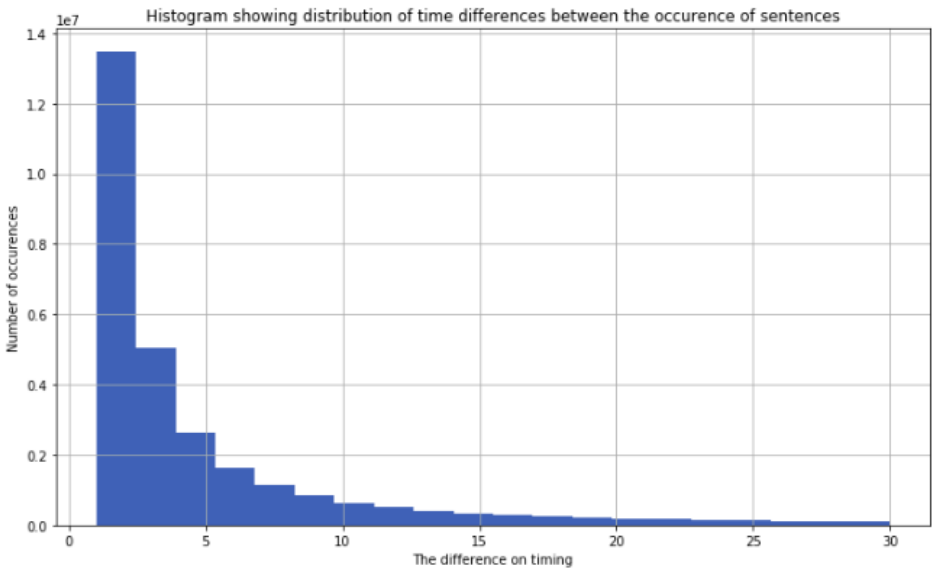

Figure 1: Distribution of Δ between the occurrence of sentences

With the following quantiles:

Quantile	**Time**
25%	1.564s
50%	2.6s
60%	3.33s
70%	4.525s
80%	6.673s
90%	11.43s

Table 2: Basic stats on French Open-subtitles data

So we chose to set $5s$ as a threshold for chunking our data. Indeed, this threshold is going to be static for the hole xml files. We manually checked the data and noticed that this threshold represent a promising bound to chunk dialogues. Furthermore, at the end of each sentence, we add a token $< SEN >$ in order to understand that we have to go to the next line and deal with the next sentence.
The following figures show us three samples of chunked and parsed French Open-subtitles data:

```
tu vas apprendre à dire ton nom .<SEN>
comment tu t'appelles ?<SEN>
kunta kinté .<SEN>
quel est ton nom ?<SEN>
mon nom ... est toby .<SEN>


neuf ans plus tard<SEN>


hé,toby !<SEN>
tu veux un peu d'eau ?<SEN>
non .<SEN>
tu es sûr ?<SEN>
tiens .<SEN>
si t'as besoin de quelque chose ... tu sais à qui t'adresser .<SEN>
et pour quelque chose dont même un africain ne peut se passer .<SEN>


négro !<SEN>
hé,toi,le négro !<SEN>
je te parle .<SEN>
va chercher quelqu'un ... pour charger cette caisse sur la charrette .<SEN>
```

Figure 2: Example of chunks

```
j'ai oubliè d'organiser la visite de ton lycée .<SEN>
papa l'a fait . j'y vais avec lui dimanche .<SEN>
j'irai le retrouver en taxi .<SEN>
il te l'a demandè ?<SEN>
c'est l'avantage de manhattan,non ?<SEN>
plus besoin de m'emmener partout .<SEN>
je t'accompagnerai .<SEN>
pourquoi venir ici si je peux pas me déplacer seule ?<SEN>
j'abandonne .<SEN>
je ne me battrai que demain .<SEN>
viens ici .<SEN>
c'est honteux de t'aimer à ce point .<SEN>
tu l'as dit .<SEN>
```

Figure 3: one chunk/dialogue

```
54  john ?<SEN>
55  john stoller ?<SEN>
56  allez-vous-en .<SEN>
57  john,avez-vous déjà eu un problème mental ... ?<SEN>
58  ce que j'essaie de vous demander c'est :<SEN>
59  avez-vous déjà séjourné dans ce service avant ?<SEN>
70  est-ce que je suis fou ?<SEN>
71  eh bien ... l'avez-vous déjà été avant jeudi ?<SEN>
72  laissez-moi tranquille .<SEN>
73  je ne crois pas que vous soyez fou .<SEN>
74  je pense savoir ce qui s'est passé .<SEN>
75  je pense savoir ce que vous avez vu ... parce que je l'ai vu aussi .<SEN>
76  la femme que vous avez attaquée ... <SEN>pas une femme .<SEN>
77  son visage,c'est horrible . c'est ça ?<SEN>
78  la décomposition,le sang,les vers . <SEN>vous l'avez vu ?<SEN>
79  oui . <SEN>il vous a touchée ?<SEN>
80  quoi ?<SEN>
81  ne le laissez pas vous toucher .<SEN>
82
```

Figure 4: one chunk/dialogue

3 Sequence to sequence model

In this section, we will try to test somehow the "quality" of our data by training a sample of our data with the seq2seq model, in order to train the model, we had to perform some preprocessing. Before digging into details, let's define what is the sequence to sequence model :

3.1 Definition

The Sequence to Sequence model (seq2seq) consists of two RNNs - an encoder and a decoder. The encoder reads the input sequence, word by word and emits a context (a function of final hidden state of encoder), which would ideally capture the essence (semantic summary) of the input sequence. Based on this context, the decoder generates the output sequence, one word at a time while looking at the context and the previous word during each timestep. This is an oversimplification, but it gives an idea of what happens in seq2seq.[3]

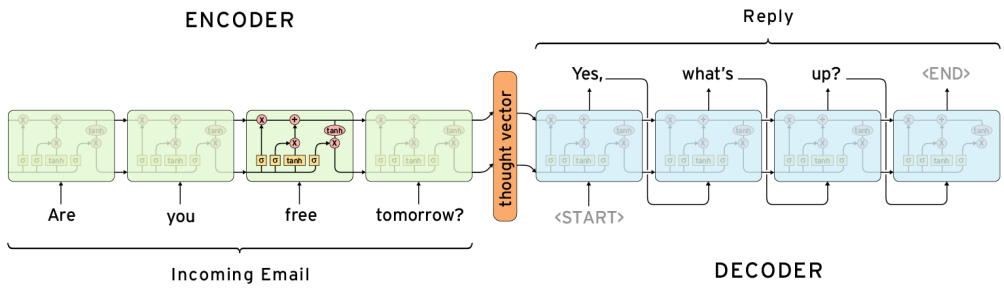

Figure 5: Encoder and Decoder RNNs

The context can be provided as the initial state of the decoder RNN or it can be connected to the hidden units at each time step. Now our objective is to jointly maximize the log probability of the output sequence conditioned on the input sequence.[3]

3.2 Pre-processing and training

The sequence to sequence model was implemented in a way that made us perform some pre-processing before feeding it. First of all, we choose to partition the sampled data -We train and test only on a sample of our data- into train/-validation/test with the following respective percentages 80%/10%/10%.

```
|---- Open Subtitles data
                            |----target.npy
                            |----encoder_input.npy
|    |---- train             |----decoder_input.npy
|                            |----enc_len.npy
|                            |----dec_len.npy
|    |---- validation       : same architecture as train
|    |---- test             : same architecture as train
|    |---- token2id.pickle
|    |---- id2token.pickle
```

Let's describe these files :

- target.npy: each line of this file contains the sentence target in the form of consecutive numbers referring to words that form the sentence (using our dictionary of vocabulary mappings) ending with SEN integer id to show the end of the sentence
- encoder˙input.npy: each line of this file contains the input sentence in the form of consecutive numbers referring to words that form the sentence (using same dictionary of vocabulary mappings).
- decoder˙input.npy: same as target.npy except it doesn't finish by SEN token. Instead, it begins by a special token GO that indicates the beginning of decoding.

- enc˙len.npy: this is a list where each element is the length of the input sentences in order.
- dec˙len.npy: this is a list where each element is the length of the target/decoder sentences in order.
- token2id.pickle: dictionary with word tokens as keys and integers as ids.
- id2token.pickle: inverse keys/values of token2id.

Our model is a single-turn sequence to sequence model. It uses SGD with Adam optimizer in learning the weights. The perplexity loss is used for validation.

The sample we chose to train/validate/test our model is:

- 55 parsed movie files randomly chosen
- Number of words: 328169
- Number of sentences: 41176
- Size of vocabulary of words: 10000
- tensor-flow randomly initialized word embedding

Once we merge all movie files in one big txt file. Our encoder/decoder inputs will have the following look:
encoder, decoder:
line1, line2
line2, line3
line3, line4
line6, line7
etc, etc
etc, etc

After preprocessing it, we run the training. The following graph shows us the decreasing training loss and validation-loss :

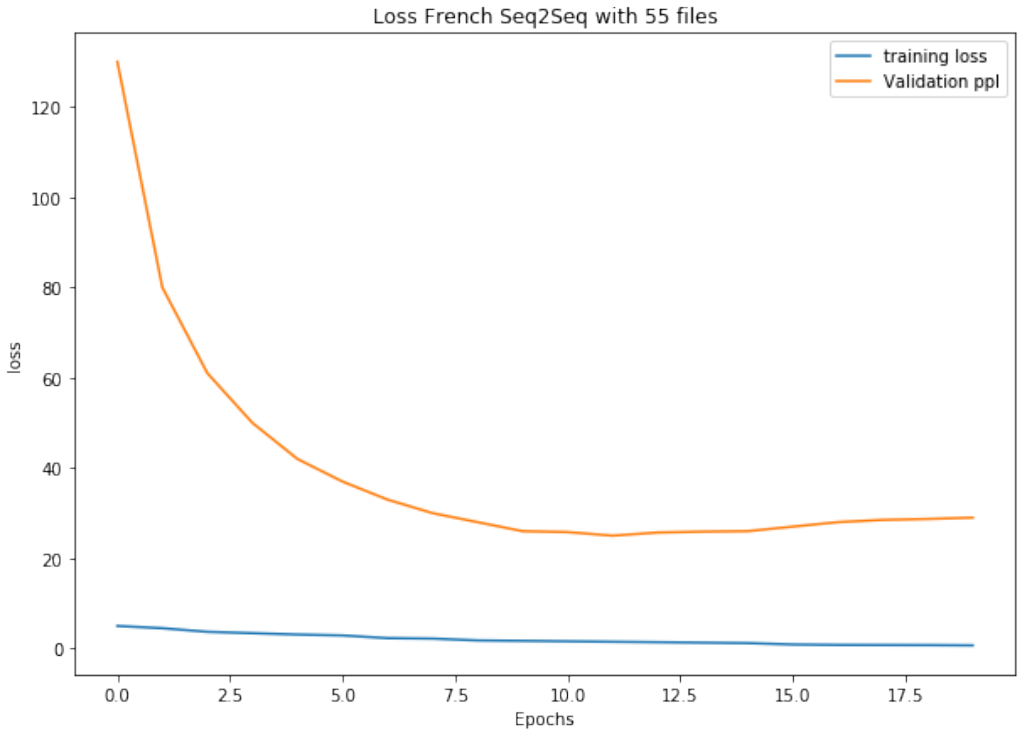

Figure 6: Training and validation loss over epochs

- the training loss decreases from 9 to 2
- the perplexity loss gets optimal to 29 within 13 epochs

3.3 Testing the model

The following table shows the query (the input sentence), the target and the prediction of our seq2seq model.

Query	Target	Prediction
vers la gauche pour défiler	escadron , en avant	très bien .
pardonne-moi,helen .	je voulais prendre le risque de te revoir .	ce n'est rien .
vous allez bien,mlle " flammèche " ?	je pensais au baron .	pas compris .
ensuite,nous leur ferons une surprise .	leur chef n'est pas avec eux .	elle est vivante .
il te faut quelqu'un comme moi . quelqu'un de ton âge .	quelqu'un qui sache comment ... te rendre heureuse .	je suis si heureuse .
on regretterait d'avoir été raisonnables et d'avoir attendu .	qu'y a-t-il de si agréable dans l'attente ?	ce n'est rien .

Table 3: Examples of queries and their targets/our model predictions

We notice that there are some cases where the model works slight okay but others where the response doesn't really make any meaningful sens.

4 Comparison between DeepMoji and LIWC

Having an emotional dictionary can be very important if we want to implement a model that uses en emotional layer in its predictions. Despite the fact that we are working on french data, we looked for English emotional dictionaries since it is much easy to find and also more trustworthy. A work of translation can be done on the English dictionary into French in order to translate sentences to its coherent emotion. First, we tested LIWC dictionary which is very known but seems to work poorly -"I am not angry" is considered to express angriness-, so we looked for an other more powerful emotional dictionary.
DeepMoji is a model trained on tweets and emojis. For each given sentence, DeepMoji gives a list of emojis expressing the embedded emotions of the sentence with confidence levels. The following chart shows us how DeepMoji is far way better than LIWC. [4]

Examples	LIWC	DEEPMOJI
I feel sick	neg-cogn	sick emoji + high confidence
I feel bad	neg-cogn	sick emoji + high confidence
I am not okay	POS-cogn	sick emoji + high confidence
Oh, super	pos	pos emoji + low conf
I am excited	pos	smiling emoji + high conf
I lost my job	neg	angry+sad emoji + low conf
I found a new job	Cognitive process	happy emoji + high conf
getting up early is exhausting	neg	sad+sick + high confidence
getting up early is	0 emotion	sad + sick + medium conf
I am on vacation	0 emotion	happy emoji + low conf
I will travel for work	0 emotion	neutral+peace+little sick + low conf
I will visit family	social words	Love + medium conf
I am angry	neg	angry emoji + high conf
I am upset	neg	SAD + ANGRY + little high
I am sad	neg	broken heart+ cry + high conf
I am happy	pos	happy + high conf
I feel loved	pos+con+social	heart + smile + medium conf
I feel appreciated	cogn+ pos	good + little high conf
I am not angry	neg + cogn	don't talk to me emoji + high conf
I am not sad	neg + cogn	happy + neutral + low conf
I am not happy	pos	sad + high conf
I don't feel loved	pos+ social	broken heart+ cry + high conf
I don't feel appreciated	pos	sad + medium conf
My flight is delayed, amazing	pos	angry + low conf
he is always watching movies	nothing	neutral + undecided + low conf
he doesn't care	pos	sad + broken heart + medium

Figure 7: Comparison of LIWC and DeepMoji emotional dictionaries

So we notice that DeepMoji is indeed better than LWIC. In fact, DeepMoji detects emotions (explicitly or implicitly), it also detects sarcasm and other "difficult-to-detect" emotions.

5 Conclusion

During this research project, several interesting topics were covered, from parsing/cleaning/searching for data to the discovery of the field of NLP and its powerful algorithms and tools. Besides the data cleaning and preprocessing part, language models with NLP neural network models and its use cases in chatbots were deeply studied through several papers.

Last but not least, working on french data is very interesting. Indeed, the way we deal with french data changes slightly compared to English data. Since data had been parsed, cleaned and analyzed. further work would be to train/tweak and improve the model on a sample first and then on the hole data using a cluster. Once this work done, what can be interesting is adding the emotion Layer using DeepMoji. Since DeepMoji only exists in English, a proposition would be to translate french sentence to English and get the output emotion of it in order to create a french emotional dictionary to be added to be integrated as a layer in the language model with the aim of detecting emotions.

References

[1] Sayan Ghosh, Mathieu Chollet, Eugene Laksana, Louis-Philippe Morency, Stefan Scherer, "Affect-LM: A Neural Language Model for Customizable Affective Text Generation"

[2] Pierre Lison, Jorg Tiedemann, "OpenSubtitles2016: Extracting Large Parallel Corpora from Movie and TV Subtitles"

[3] Oriol Vinyals @Google, Quoc V. Le @Google, "A Neural Conversational Model"

[4] Bjarke Felbo, Alan Mislove, Anders Søgaard, Iyad Rahwan, Sune Lehmann, "Using millions of emoji occurrences to learn any-domain representations for detecting sentiment, emotion and sarcasm" @MIT