Published in Analytics Vidhya · Follow

Fatima-Ezzahra Fettah · Follow
Mar 17, 2020 · 16 min read

Abstractive Text Summarization

Sometimes, we need concise information in a given document rather than too much details…

Introduction:

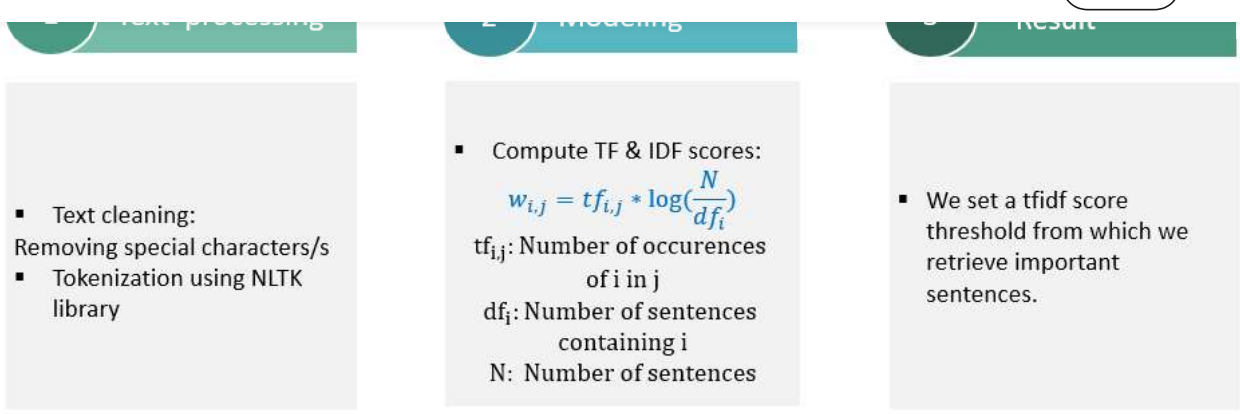
One of the challenges in natural language processing and understanding is text generation that could be applied for example in text summarization. In fact, it consists of understanding the information conveyed by a given text and reducing its size into a concise summary that contains the main important and relevant information. In this context, several studies are conducted to build extractive summaries based on frequentist approach. The main idea of those algorithms is to set a scoring system that allocates high values to the sentences assumed important and vice versa. However, the complexity lies in conceptualizing a model that goes trough the whole text, keep the important information in its memory and skip noisy ones like redundancies, details… In short, the model must create a context vector from a long text.

1st Approach: TFIDF Summarizer

It is obvious that a model performs well on a concentrated text that contains just concise information with less noise. In this context, we can extract important parts from a given text using some extractive summarizers that will be the input of the summarization model. I chose to work with TFIDF vectoriser to reduce the size of the text and get the most important sentences based on the following scoring system. To test the model, we chose a text about “Artificial intelligence” from Wikipedia to see how the model will perform since it will be the input of the summarization model.

$$TFIDF_{t,d,D} = TF_{t,d} * IDF_{t,D}$$

Importance of term t in a sentence d = Frequency of term t in a sentence d * Importance of term t in the set of sentences D



While reading the summary generated, we can notice that there is not a coherence between sentences knowing that they will be fed into the abstractive summarization model subsequently built.. Therefore, if we let ourselves be done, we introduce biased data into our model. This means that our model has to get the context vector from an incomprehensible and non-homogeneous text. Consequently, we will get biased summary.

Change of approach:

How does a human being generate a summary?

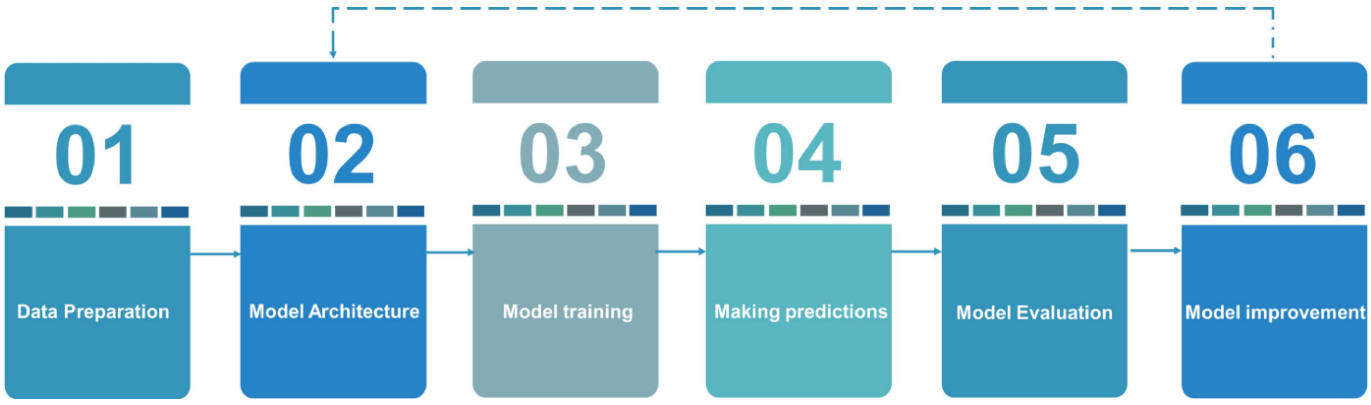
- If you are asked to write a summary of a scientific article for example, the first step you will do is to read the full text and after that you will summarize each part separately. Therefore, we can overcome the problem of text size and assume that we will summarize the text by a step of 10 rows.

How my algorithm could be applied in business?

- Clients reviews can often be long and descriptive and you can't imagine the time and effort consumed to analyse and get the main point from those reviews. In this context, we are facing a Natural Language Understanding problem. How can we generate short and brief summaries for those long reviews?

Case of Study: Amazon Customers Reviews

- As a case study of my project, I chose to work on Food Reviews provided by Amazon.The objective is to generate a summary for the Amazon Fine Food reviews using the abstraction summary approach based mainly on Keras library.
- Below the pipeline of the project:



Implementing Text Summarization in Python using Keras

A- Data Preparation:

As mentioned before, the dataset consists of Amazon customers reviews. It contains about 500000 reviews with their summaries which required a huge capacity of training. Limited by this constraint,we fix the number of rows to 100000 assuming that it will be

```
> Convert the reviews and summaries to lowercase
> Remove HTML tags
> Contraction mapping: that consists of importing a contraction dictionary for example: didn't becomes did
not etc...
> Remove ('s)
> Remove any text inside the parenthesis ( )
> Eliminate punctuation and special characters
> Eliminate stop words
> Eliminate short words
```

Below some samples of clients reviews and their summaries after the preprocessing. As you can notice, the summaries starts and ends with -Start- and -END- tokens respectively as a limitations of their beginings and their ends.

```
Review: bought several vitality canned dog food products found good quality product looks like stew
processed meat smells better labrador finicky appreciates product better
Summary: _START_ good quality dog food  _END_

Review: product arrived labeled jumbo salted peanuts peanuts actually small sized unsalted sure error
vendor intended represent product jumbo
Summary: _START_ not as advertised  _END_

Review: confection around centuries light pillowy citrus gelatin nuts case filberts cut tiny squares
liberally coated powdered sugar tiny mouthful heaven chewy flavorful highly recommend yummy treat familiar
story lewis lion witch wardrobe treat seduces edmund selling brother sisters witch
Summary: _START_ delight says it all  _END_

Review: looking secret ingredient robitussin believe found got addition root beer extract ordered made
cherry soda flavor medicinal
Summary: _START_ cough medicine  _END_

Review: great taffy great price wide assortment yummy taffy delivery quick taffy lover deal
Summary: _START_ great taffy  _END_
```

B- Modeling:

Model Architecture:

After cleaning the data, we proceed to the modeling part that consists of building an analytics model taking the tokenized data.

i. Reviews & Summaries Tokenization:

For text tokenization, we use Keras tokenizer via 3 functions:

1. **fit_on_texts:**

Updates internal vocabulary based on a list of texts. This method creates the vocabulary index based on word frequency. So, if you give it something like “The cat sat on the mat.” It will create a dictionary : word_index[“the”] = 1; word_index[“cat”] = 2.

It is word — index dictionary so every word gets a unique integer value. 0 is reserved for padding. Lower integer means more frequent word (often the first few are stop words because they appear a lot).

2. texts_to_sequences:

Transforms each text in texts to a sequence of integers. So it basically takes each word in the text and replaces it with its corresponding integer value from the word_index dictionary.

We take as an example: [“The earth is an awesome place live”,”Earth protection is our common goal”]

- This is just an example to show the output of functions listed above. As we can notice, the sequences have different sizes so it can not be fed into the models we are going to conceptualize. To do so, we use padding technique that consists of adding zeros to have the same sequences length.

3. pad_sequences function: Pads sequences to the same length.

This function transforms a list of num_samples sequences (lists of integers) into a 2D Numpy array of shape (num_samples, num_timesteps). num_timesteps is either the maximum length argument if provided, or the length of the longest sequence otherwise.

- Sequences that are shorter than num_timesteps are padded with value at the end.
- Sequences longer than num_timesteps are truncated so that they fit the desired length.

The position where padding or truncation happens is determined by the arguments padding and truncating, respectively.

ii. Seq2Seq: Sequence to Sequence model

1. **Theoretical Approach**

We are going to implement seq2seq model on the reviews&summaries preprocessed before. Seq2seq is a deep learning model that converts an input sequence into an output sequence. Our input and output are denoted by X and Y respectively; X refers to the customer Review and Y refers to its summary.

$$X = [x_1, x_2, \ldots, x_i, \ldots x_n]$$
$$Y = [y_1, y_2, \ldots, y_j, \ldots y_m]$$

- And let the vocabulary of input and output be represented as following: V^I, V^O. We can assume that V^O est inclus dans V^I.

The purpose of seq2seq is to understand and model the probability of generating the output Y when the input sequence X is given. Since the model generates word by word the conditional probability modeling could be seen as the probability of generating the jth word when a set of words belonging to the summary and an input were given;

$$P(Y/X) = \prod_{j=1}^{J+1} P_\theta(y_j/Y_{<j}, X)$$

The seq2seq consists of 2 processing steps:

a- Generating the fixed size vector Z from the given input sequence X: Z=f(X)

- f could be any recurrent neural network such as RNN, LSTM…

b- Generating the summary output from the fixed size vector Z.

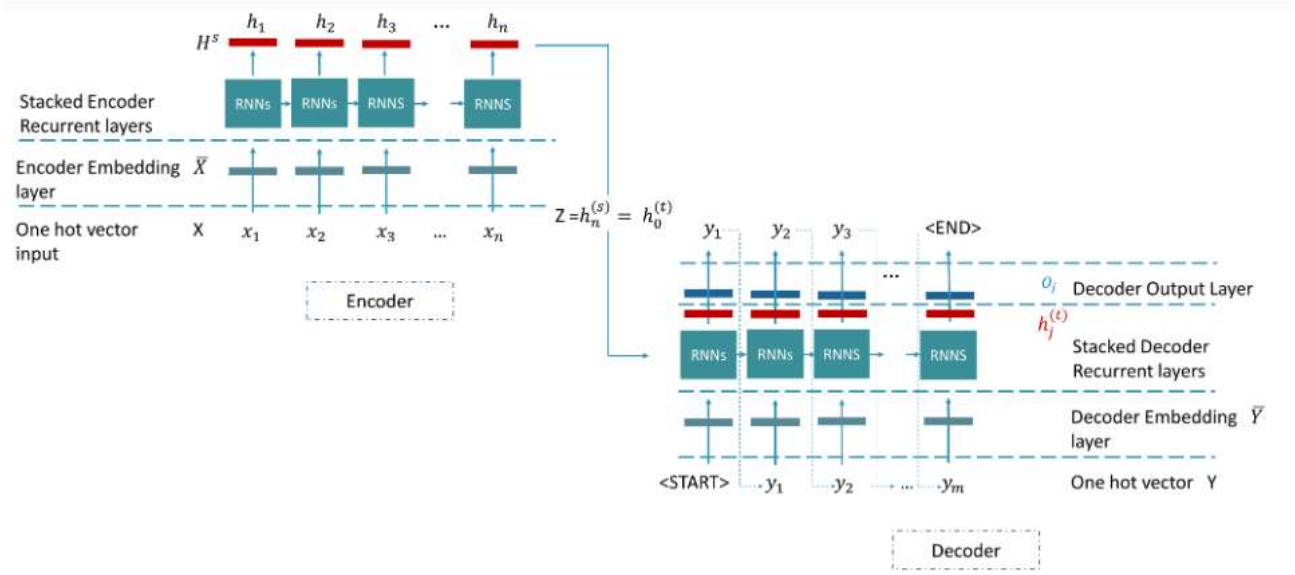
- In fact, the probability $P(y_j/Y_{<j}, X)$ is computed via $P(y_j/Y_{<j}, Z)$

$$P(y_j/Y_{<j}, Z) = g(h_j^t, y_j), h_j^t = K(h_j - 1^t, y_j - 1)$$

K: the function that generates the hidden vector hj

g: the function to calculate the generative probability of the one-hot vector yj which justifies the recurrent aspect mentioned above.

2. Model Architecture:



The encoder consists of two layers: the embedding layer and the recurrent layer(s), and the decoder consists of three layers: the embedding layer, the recurrent layer(s), and the output layer.

Encoder Embedding Layer:

- The embedding layer converts each word of the input sequence into an embedding vector.

x_i: is a one hot vector that represents the i-th input word.

$\bar{x}_i$: is the embedding vector of the i-th input word $\bar{x}_i = E^s x_i$

Where: $E^s \in R^{D*|V^s|}$ is the embedding matrix of the encoder.

Encoder Recurrent Layer:

The encoder Recurrent layer generates the hidden state *hi* from the embedding vectors: F refers to an activation function often taken non-linear (sigmoid, tanh…)

$$h_i = F(W^s * \begin{bmatrix} h_{i-1} \\ \bar{x}_i \end{bmatrix} + b^s)$$

Decoder Embedding Layer:

The decoder recurrent layer converts each word in the output sequence into an embedding vector

$$\bar{y}_j; \bar{y}_j = E^t y_j \text{ Where: } E^t \in R^{D*|V^t|}$$

Decoder Recurrent Layer:

The decoder recurrent layer generates the hidden vectors from the embedding vectors.

$$h^t_j = G(W^t * \begin{bmatrix} h^t_{j-1} \\ \bar{y}_j \end{bmatrix} + b^t)$$

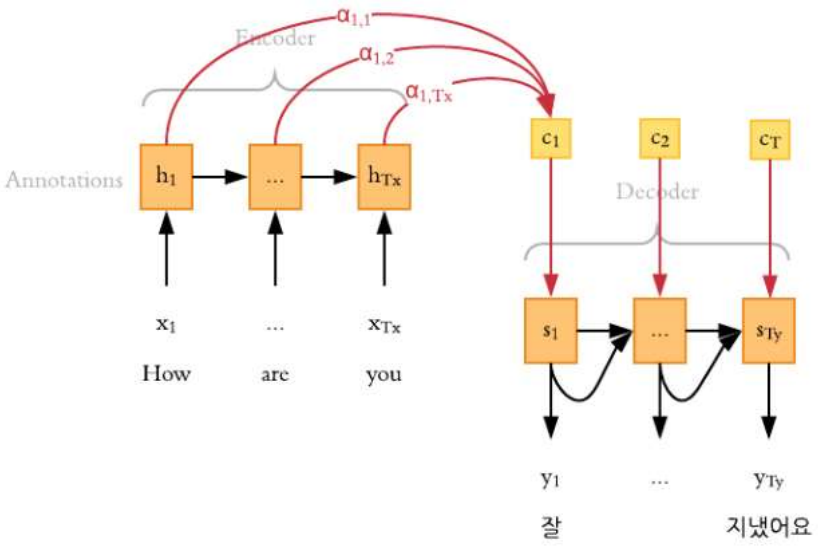
Decoder Output Layer:

The decoder output layer generates the probability of the j-th word of the output sentence from the hidden vector.

Attention mechanism: Bahdanau attention

After feeding the input sequences into the encoder part, the model has to recognize all the important parts of the text and skip all noisy information including redundancy.

Consequently, “Attention” is an interface between the encoder and decoder that provides the decoder with information from every encoder hidden state. With this setting, the model is able to selectively focus on useful parts of the input sequence and hence, learn the alignment between them. This helps the model to cope effectively with long input sentences.



Encoder works as usual, and the difference is only on the decoder’s part. The decoder’s hidden state is computed with a context vector, the previous output and the previous hidden state. But now we use not a single context vector c, but a separate context vector c_i for each target word.

RMSPROP optimizer:

In order to optimize the cost function of our seq2seq model, we opted for the RMSPROP optimizer for those reasons

- It tries to adjust the learning rate and does it automatically.
- It chooses different learning rate for each parameter; which is in our case layers weights.

This update of weight is done seperatly as follows:

$$v_t = \rho . v_{t-1} + (1 - \rho) . grad_t^2$$

$$\Delta w_t = \frac{\eta}{\sqrt{v_t + \epsilon}} . grad_t$$

$$w_{t+1} = w_t + \Delta w_t$$

Where:

η: Initial learning rate

vt: Exponential average of squares of gradients

gt: Gradient at time t along wi

the model does not improve we stop turning the model.

The training phase consists of setting the weights of layers that allow us make an accurate predictions on an unseen reviews by inference. Below, the predicted summaries we generate on new customers reviews:

```
Review: ordered salmon thursday january received january salmon delicious wooden box nice design used
store items future
Original summary: alaska smokehouse smoked salmon
Predicted summary:  great deal


Review: drank cold could pleased coffee high quality arabica always notice coffee arabica robusta sweet
without sweet mean cannot compare major bottled coffee brand reason alone keep fridge work offer people
without looking like pushing calories along caffeine long time would stock highly sweetened coffee
beverages fridge work noticed people stopped consuming happened around time lost interest level sweetness
recommend drink prefer lower level sugar old school arabica coffee types
Original summary: it was perfect little sweet without being too sweet
Predicted summary:  great coffee


Review: variety granola one ranks top favorites crunch granola coconut main taste sweet overly eat snack
craving something fills fast satisfies sugar cravings
Original summary: great taste
Predicted summary:  yummy


Review: bought dollar tree yummy taste like french fries crunchy going buy bulk pass kid test
Original summary: yum
Predicted summary:  great snack


Review: used eating flaxseed brownie hodgson mill brownies super easy make taste great since like dark
chocolate usually add little cocoa
Original summary: delicious brownie
Predicted summary:  great gluten free bread


Review: picked local grocery store organic brown rice pleasantly surprised cooks really well sticky tastes
great typically brown rice lover kind really good kids husband ate asked seconds well delighted see amazon
great price subscribe save
Original summary: love this rice
Predicted summary:  great rice


Review: stuff oily probably could run car takes cup creamer make change black tan ick
Original summary: yuk
Predicted summary:  not so good


Review: good concentrate product packaging recyclable good taste rather unripe taste liking
Original summary: good not so good
Predicted summary:  not bad


Review: cream instead compare office coffee cafe cube farm definitely matches taste quality easy quick
mellow great idea soul wants cup coffee cannot bear brew one two lonely cups big disadvantage instant
coffee far concerned serving product abut caffeine compared caffeine brewed cup coffee want caffeine
flavored versions product even less caffeine per serving although could tell disadvantage suppose whole
house fill sound fragrance brewing coffee choose instant alas still given convenience decency beverage
buying little single sticks pretty regularly recommended
Original summary: pleasant beverage convenient format
Predicted summary:  good coffee


Review: price right comparison local arts store steal color dark picture pastel red set pink anything
reccomended measurement usage delivered bubble wrap package
Original summary: red or pink
Predicted summary:  great candy


Review: crumby grapes sweet enough wine good excuse press bad news things rediscovered many busy helping
image culinary california australia everybody getting act thin flabby faux hitting shelves cute backbone
courage heavy lifting kitchen might well pour lemonade grilled salmon acidic glory taste alone compare
anything might run across vineyards make veritable next try gently fresh really ripe fruit loves swirled
```




Review: huge tea fan one favorites think taste subtle honey compliments well next time add lemon slices maybe orange slice yummy
Original summary: very nice tea
Predicted summary: very good

Review: price excellent heard many good things kind bars mango macadamia bar taste great would willing try flavors would buy flavor
Original summary: they did not taste that great
Predicted summary: good but not great

Review: used mcp pectin probably years making freezer jam jam always sets looks tastes perfect hard time finding mcp recently forced try another product making jam last summer jam set unhappy result thrilled find mcp amazon supply last couple years convinced mcp best wish could buy locally grocery store
Original summary: mcp is the best
Predicted summary: the best

Review: little guy ate could even write review
Original summary: great deal
Predicted summary: not my favorite

Review: like cashews way biased really enjoyed earn name salty though product need eat low sodium rest tasty treat
Original summary: tasty treat
Predicted summary: good but salty

Review: younger prone trying whatever new product came along various benefits seeking seldom ever payoff see advertisement product promising bigger better stronger even consider imagine surprise tried new shampoo actually noticed volume hair blow drying even two people asked would hair cut day used seriously miraculous change shampoo weave definitely makes noticeable difference negative comment fairly strong perfume scent tend avoid soaps shampoos absolutely continue use
Original summary: unbelievable literally
Predicted summary: good but not the best

Review: item allows convenience safe place emergency funds hides well pantry know whenever need
Original summary: safe safe
Predicted summary: good for the price

Review: mostly enjoyed paul newman donut shop ones one day saw count twenty bucks thought hmmm good deal must awful heck try best coffee ever tasted never see searched online hot simply poured ice hands favorite cannot drink constantly recommending people agree hooked also great environmentally packaged bet lack plastic makes taste better cannot lose drink two three cups day sip good first fact going make cup right san francisco bay coffee organic one cup keurig cup brewers rainforest blend count
Original summary: best coffee ever
Predicted summary: best tasting coffee have ever had

Review: love product fact become addictive time however noticed hard pieces chewing nuts examining closely think somehow shells got nuts shelled made careful bitterness otherwise wonderful flavor also want crack chip tooth hope isolated instance batch
Original summary: emerald dry roasted walnuts review
Predicted summary: not so good

Review: reading reviews decided purchase husband husband drinks tea every day prefers stronger blends like one used drink tips tried barry tea prefers barry looking nice strong basic tea look
Original summary: my husband favorite everyday tea
Predicted summary: great tasting green tea

Review: rich man peanut butter cap crunch poor man honey nut cheerios tastes like former looks like latter wife dug one evening hours later demolished peanut except artificial peanut flavor looking sugar brand names satisfy craving nom nom
Original summary: sugary nut crunch
Predicted summary: not so good

Review: eat gluten free tried almost every type bar available pleasantly surprised got try bar moist delicious tasted like blueberry lemon cookie big bursts blueberries cannot wait try flavors thanks pamela making another great tasting product




predicted ones.

Model Evaluation: . Semantic similarity evaluation:

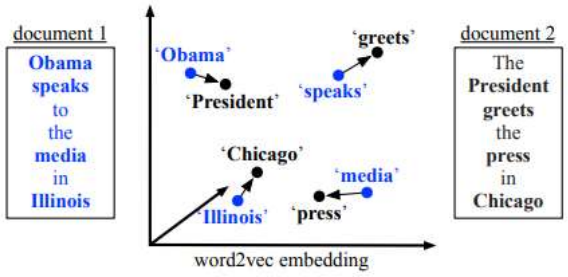
In this part, we will evaluate the quality of our predictions using a semantic similarity approach which means we will quantify the meaning similarity between the predicted and the original summary using pretrained word2vec model.

- In this context, we worked with pre-trained word embedding model 'Word2vec' trained on Google News to get the vector representations for each summary's word either in the original and the predicted one. Therefore, we are facing a problem of similarity computing presented as follows:

Suppose we want to calculate the similarity between those two sentences:

S1 = 'obama speaks media illinois' and S2 = 'president greets press chicago'

It is true that those sentences don't share the same words but maybe they conveyed the same information. At this stage, each sentence is represented as a matrix but the question that arises how could you make a match between the similar words belonging to each sentence. For example, How could you associate Obama to president? (Whis is represented in the image below)



Captured from Kusner et al. publication

Consequently, we use Word Moving Distance optimization algorithm inspired from the usual optimization problem Earth Mover Distance. It allows transfer every word from sentence 1 to sentence 2 because algorithm does not know "obama" should transfer to "president". At the end it will choose the minimum transportation cost to transport every word from sentence 1 to sentence 2.

- WMD algorithm:

We assume that we have a vocabulary of n words {1, . . . , n} and a collection of documents. We assume that the set of distinct words of D and D' are, respectively, $\{w1,\ldots,w|D|\}$ and $\{w1,\ldots,w|D'|\}$. Moreover, we use Di to denote the normalized frequency of wi, that is, the number of occurrences of wi in D over the total number of words in D. We use $D'j$ to refer to the normalized frequency of $w'j$ analogously. Note that:

$$\sum_i D_i = \sum_j D'_j = 1$$

xi denotes the embedding of a word i in a vector space of dimension d and c(i,j) denotes the Euclidean distance between the embedding of words i and j, that is,

$$c(i, j) = ||x(i) - x(j)||_2$$

- D: John likes algorithms. Mary likes algorithms too
- D': John also likes data structures.

Our vocabulary: V = {1:'John', 2:'likes', 3:'algorithms', 4:'Mary', 5:'too', 6:'also', 7:'data, 8:'structures'}

The sets of distinct words of D and D' are:

Therefore, the normalized documents representations are:

```
D = {1/7,2/7,2/7,1/7,1/7}
D'= {1/5,1/5,1/5,1/5,1/5}
```

- Problem formulation:

$$\min_{\mathbf{T}\geq 0} \sum_{i,j=1}^{n} \mathbf{T}_{ij} c(i,j)$$

$$\text{subject to: } \sum_{j=1}^{n} \mathbf{T}_{ij} = d_i \quad \forall i \in \{1,\ldots,n\}$$

$$\sum_{i=1}^{n} \mathbf{T}_{ij} = d'_j \quad \forall j \in \{1,\ldots,n\}.$$

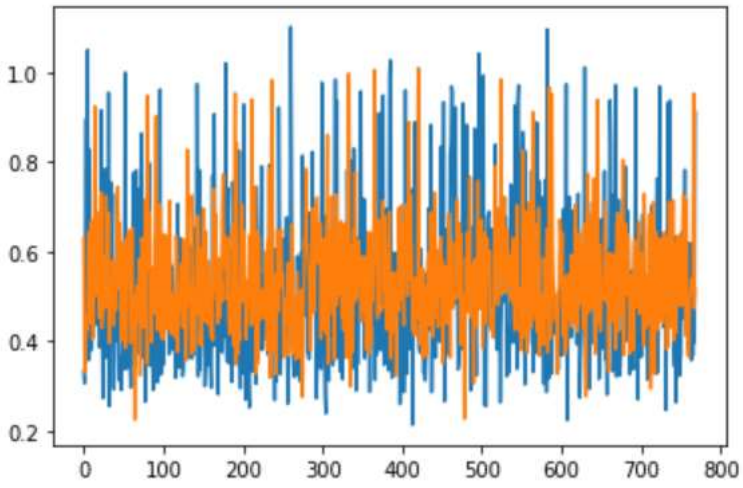
Captured from Kusner et al. publication

Where: c(i,j) the cost of transporting word i to word j.

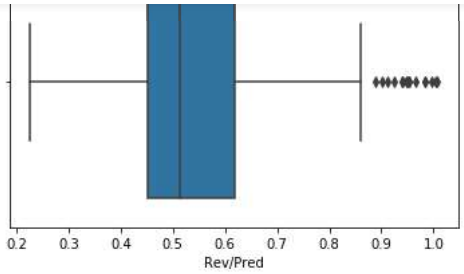
$$c(i,j) = c(w_i, w'_j) = \begin{cases} 0 & \text{if } w_i = w'_j \\ ||x(w'_j) - x(w_i)||_2 & \text{if } (w_i, w'_j) \in C \\ c_{max} & \text{otherwise} \end{cases}$$

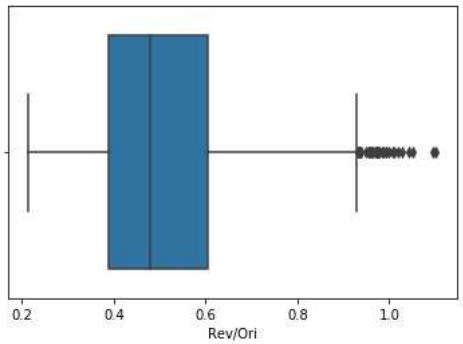
After loading the pretrained vectors and solve the optimization problem that finally gives the semantic similarity between each predicted and original summary and its reviews.

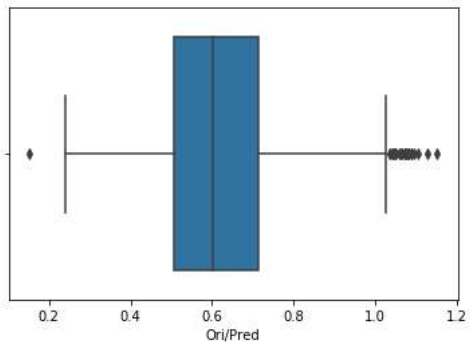
The variations of semantic similarity between the reviews and their original and predicted summaries are shown in blue and orange respectively. Therefore, we can notice the two quantities vary identically.



If we are interested in the semantic similarity between reviews and their summaries; original & predicted. We can see in the box plots below that those two quantities have almost the same distribution in term. In fact, 50% of the predicted summaries are 52% semantically similar to their reviews. On the other hand, 50% of original summaries are assumed to be 48% similar in terms of meaning to their own reviews.







Conclusion:

The summaries generated by seq2seq model and all the techniques listed above contain the product the customer is reviewing and his judgement in a concise and short sentence. In fact, the model is able to go through the entire text, keeps important information notably the product and the judgement and skip all noisy details using recurrent neural networks such us LSTMs. However, sometimes the model generates biased summaries especially when customers start comparing the product with an other one that does the same function or a substitute. In that case, the model makes an error while detecting the main product the customer is reviewing and the judgement as well. Therefore, we can simply overcome this problem by training the model on more data and question the assumptions already made especially in the data preprocessing step.

The python implementation of the project is available in the following Github repository: https://github.com/Fatima-EzzahraFettah/AbstractivesSummarizationSeq2seq