# Introduction:

## Context and motivation:

In this report, we summarize the different research efforts and experiments that we conducted on ALTEGRAD challenge, which is hosted on Kaggle, concerning the prediction of the *h*-index of research papers authors. We are dealing with a regression problem using texts of articles abstracts and a graph network that matches the authors that co-authored the same papers. The target variable of this problem is the h-index which can be defined as the maximum number of published papers h that have each been cited at least h times. This metric measures the performance of an author with regards to the number of citations of his publications. During the challenge, we worked on a large-scale dataset, and followed the regression problem pipeline, that is we went from cleaning and preprocessing the data, features engineering to choosing the right regression models and tuning their parameters, finally measuring their performances to choose the best ones for the competition.

## Evaluation Metric:

When it comes to measuring the performance of our models, the challenge asses the performance of our models using the **mean absolute error (MAE)**, which is the average of the absolute errors over the dataset. We can express it as:

Where N is the number of observations in the evaluated dataset, the predicted value (h-index) for the ith observation and its actual values. During this competition, we are trying to reduce the errors our prediction, thus we are looking for small MAE values. Finally, after evaluation our model on our train and validation sets, we predict on tests dataset, which we do not know its h-index values, and then submit the .csv file into Kaggle the get a real evaluation of our performance.

# Text Preprocessing:

As we are dealing with papers abstracts which represent text data, we need to clean it from white spaces, punctuations, stops words and to preprocess it so we can have same representation of the same word, as example for the verb “write” we can find many formats like “writes”, “writing” so we need to normalize it to the same original word and thus have at the end the same vectorial representation of that word. This helps us build a strong corpus that represents main vocabularies of the abstracts and reduce the noise in the text representations. The provided data, “abstracts.txt”, is a txt file where each paper ID is matched with an inverted index which is a dictionary where the keys are the words, and their values are lists of their corresponding positions in the abstracts. We can summarize this part of data preparation in the following steps:

* Read text from files using “utf-8” encoding to get almost all words present in the abstracts
* Loop over each paper id and extract the words in its inverted index.
* We check if the words are alphabetic and not stop words and if they are not empty.
* If true we change the word to lowercase and we lemmatize it. Lemmatization consists of the changing the inflected form of a word to its original format (as explained in example above with the verb write) so it can be analyzed as a single vocabulary. To do so, we used the ***WordNetLemmatizer***from the ***nltk*** package. This lemmatizer is a pretrained model that inputs the word and its tagging (verb, noun, adjective and adverb) so it can find a meaningful base form based on the context. To get the tag of a word, we built a function ***get\_wordnet\_pos()*** that uses ***nltk.pos\_tag***, a pretrained tagging model from ***nltk*** package, and output a wordnet tagging as a parameter for our ***WordNetLemmatizer.***
* Now that we have our lemmatized word, we put it in its right positions in our abstract using the inverted index values.
* Finally, we output a dictionary with papers ID as keys and list of processed words in their right positions in the abstract.