INF554 oral presentation

Kaggle group **B3**
Bichot Lilian
Bienvenu Julien
Bienvenu Marie

December 15th 2021

Contents

Introduction

Brief overview of the problem :

- Estimating h-indexes given the co-authoring graph and a bundle of abstracts

Our method :

- A lot of discussion on the relevance of each feature and its meaning
- Custom made features and predictors
- Parsimony, running programs locally

k70

Main algorithm : gradient boosting regressor on well-chosen features

Features :

- 4 Graph-based
- NoP
- Word analysis feature(s)

Graph-based : 4 features

Degree : Number of co-authors for a given author

Core-number : Quantifies the density of the co-authoring graph around an author

Eigenvector centrality (log-scale) : Measure of the influence an author has on the co-authoring network (prestige score)

PageRank : Ranking of the authors based on the structure of the incoming links in the coauthoring-network. It was originally designed as an algorithm to rank web pages

Number of Papers

- Custom-made feature
- Number of abstracts for a given author in our data
- Between 0 and 5

Remarks : If NoP < 5 then
H-index ⩽ NoP
(post-processing possibilities)

Word Analysis feature(s) : Gensim+GloVe

GloVe : Unsupervised learning algorithm for obtaining vector representations for words

- Training set : Wikipedia 2014 + Giga word 5

Gensim : Powerful library to use and train word embeddings

Word Analysis features

- Means of the word2vec (size 300) of every word used by an author
- Gives us 300 features (or less with PCA, e.g 5 features MSE 77)
- Weights : proximity to non discriminating words

Fitting/Regression algorithm

Used : **Gradient Boosting Regressor**

Tried : LASSO and RIDGE

Could have used :

- Neural networks
- Random forest
- Support vector machines

Fitting/Regression algorithm

Used : **Gradient Boosting Regressor**

Tried : LASSO and RIDGE

Could have used :

- Neural networks
- Random forest
- Support vector machines

An alternative : Deep learning

- higher potential
- more difficult to understand what the machine understands
- how do we represent words in a numerical word ?
- how do we design a neural network for H-index regression ?

BERT's word tokenizer (Bidirectional Encoder Representations from Transformers)

"don't be so judgmental"

BERT's tokenizer : splits according to semantic meaning

['don', "'", 't', 'be', 'so', 'judgment', '##al']

converting to numeric IDS

[2123, 2102, 2022, 2061, 8689, 2389]

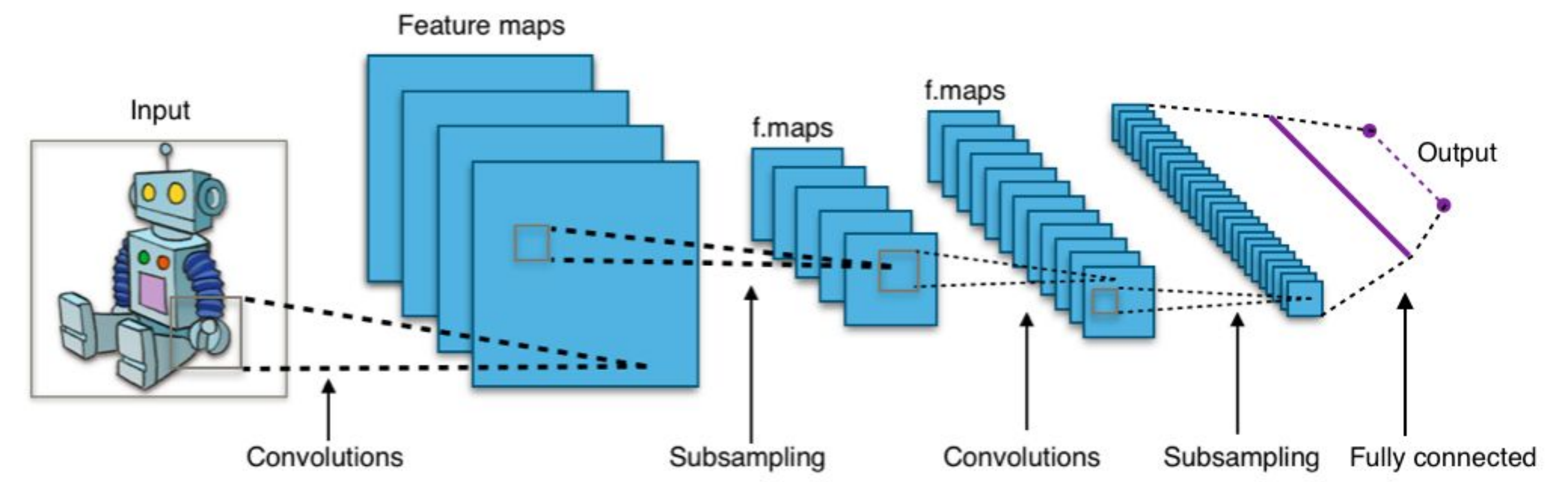
BERT's constraints

- The tokenized version of a sentence is variable in size
- The output values are integers
- The order of the words has to be used (words are not independent)

How are we going to tackle those limitations with our neural network ?

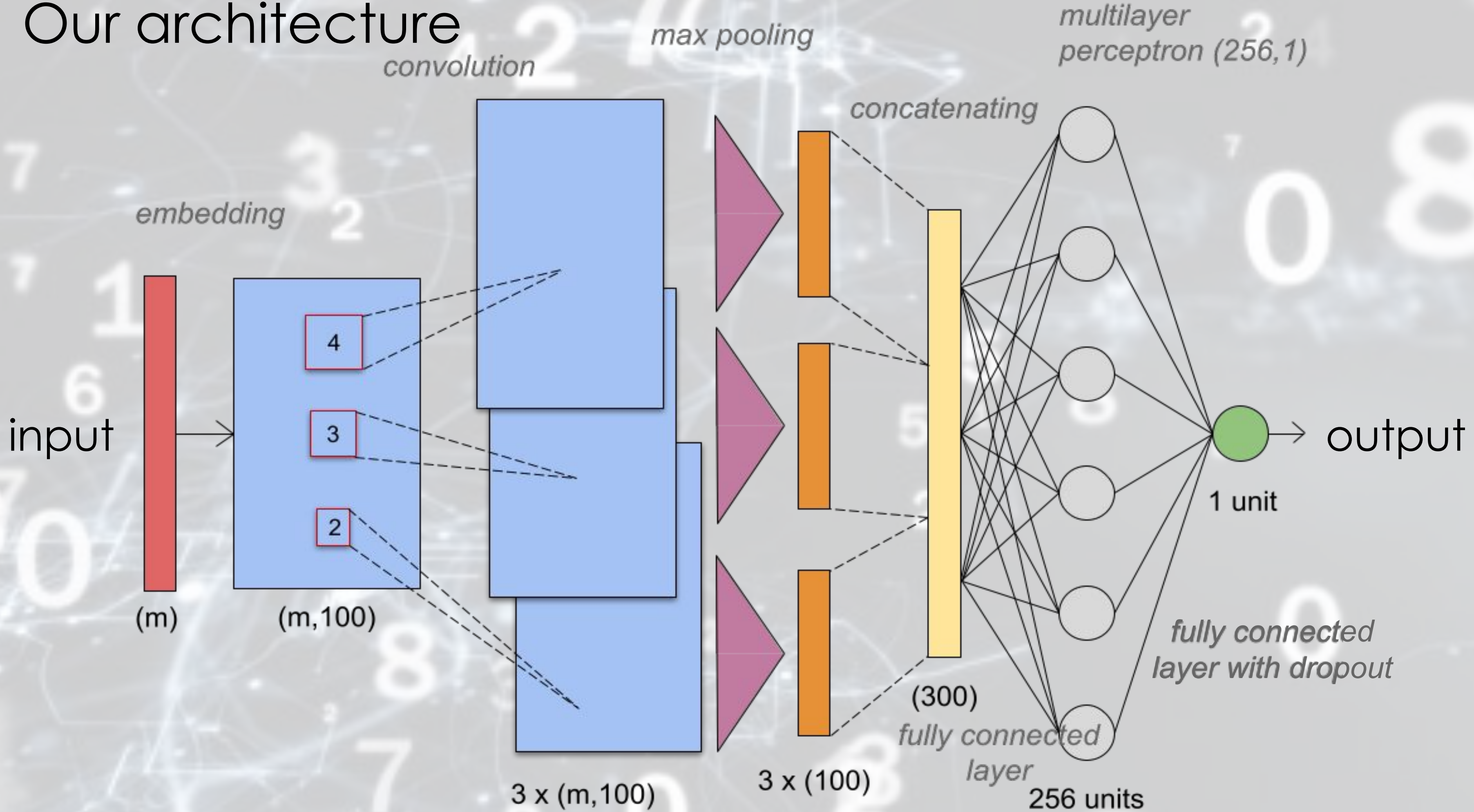
- Input are indexes : embedding
- Input of variable size : max pooling
- Order in the sequence of inputs : convolutional layers

Convolutional Neural Network : classic architecture



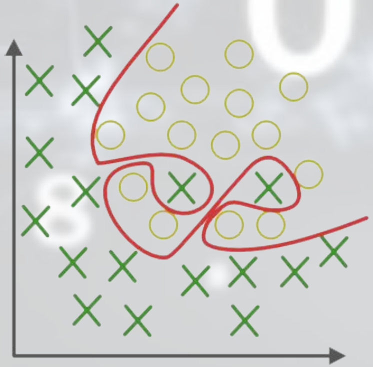
https://en.wikipedia.org/wiki/Convolutional_neural_network#/media/File:Typical_cnn.png

Our architecture



Focus on the dropout rate and overfitting

- the idea is to randomly ignore units from the hidden layers during training (trimming connectivities)
  - exclude rare dependencies
  - reduce overfitting tendencies



- the dropout rate can be interpreted as the probability for a unit to be ignored

Results

- A bit disappointed in the results of our networks
- Very good score on the train set, and bad on the test set
  - overfitting
  - attempts : heightening the dropout rate, limiting the number of learning steps
- Good potential but hard to unlock it
  - time-consuming
    - Google Colab’s GPUs

Another predictor : **analytical predictor**

k138

Description :

- $(\text{Green}_1, \text{Green}_2, \ldots, \text{Green}_k) \mapsto$ h-index (Red)
- For instance, **mean** of the partial h-indexes of the words Red uses, partial h-index defined as the **mean** of the greens that use this word

Score interpreting :

- Do **not** take into account the co-presence of words
- **Noise** due to non-discriminative words

Extension :

- Fit the partial H-indexes with a stochastic approach
- Smarter function, with a parameter we **optimize** in the training set

Another predictor : **mean of neighbors**

k147

Description : mean of green neighbors if the red has some, otherwise mean of all greens

Score interpreting : since big h-indexes are more connected, prediction **too high**

Extension : smarter function

- **depth**
- weighted mean
- functions of h-indexes with a parameter we **optimize** in the training set

Graph-based extensions

	With clustering	**Without** clustering
Coauthorship graph	Function of the greens	
Graph of **words**	Function of the greens in the cluster that has the more words	Features : functions of functions from **networkX** on the words

Thank you for listening :)