Industrial project 234313

Serverless for big data processing in the cloud (IBM)

**Project report**

Written by:

**Ido Yehezkel & Ohad Zohar**

Supervisor:

**Gil Vernik**

Contents

[Hyperparameters optimization 3](#_Toc534129412)

[Introductions and Motivation 3](#_Toc534129413)

[Facebook fastText 3](#_Toc534129414)

[Phase A - fastText prediction on the cloud using IBM-PyWren 3](#_Toc534129415)

[Comparing local prediction vs. IBM cloud function prediction 4](#_Toc534129416)

[Phase B – training and testing a fastText model 5](#_Toc534129417)

[Comparing local training vs. IBM cloud functions training 5](#_Toc534129418)

[Concl 6](#_Toc534129419)

# Hyperparameters optimization

## Introductions and Motivation

In machine learning, hyperparameters optimization and tuning is the problem of choosing a set of optimal hyperparameters for a learning algorithm. This problem is unique to each learning algorithm and there is no absolute solution. In this section of our project we show that using IBM-PyWren and IBM cloud functions one can optimize the hyperparameters in an easy and fast way. This section is not about how to choose the right optimization algorithm (i.e. how to choose the next parameters values). It’s about building a utility for distribution and parallelism over IBM-PyWren while using any algorithm that comes to mind.

## Facebook fastText

fastText is a library for learning of word embeddings and text classification created by Facebook's AI Research (FAIR) lab.

Deep neural networks have recently become very popular for text processing. While these models achieve very good performance in limited laboratory practice, they can be slow to train and test, which limits their use on very large datasets.

fastText helps solve this problem. To be efficient on datasets with very large number of categories, it uses a hierarchical classifier instead of a flat structure, in which the different categories are organized in a tree. This reduces the time complexities of training and testing text classifiers from linear to logarithmic with respect to the number of classes. FastText also exploits the fact that classes are imbalanced (some classes appearing more often than other) by using the Huffman algorithm to build the tree used to represent categories. The depth in the tree of very frequent categories is therefore smaller than for infrequent ones, leading to further computational efficiency.

We decided to use fastText as our learning algorithm because it is easy to use, fast, the data is textual and simple to deploy.

<https://en.wikipedia.org/wiki/FastText>

<https://research.fb.com/fasttext/>

# Phase A

## fastText prediction on the cloud using IBM-PyWren

After choosing fastText as our learning algorithm we tried running it for the first time over the cloud. We used a trained fastText models (downloaded from fastText git), copied it to a docker and deployed that docker to the cloud as an action. The models and datasets we used were AGnews, Amazon review polarity, DBpedia, Sogou news and Yelp reviews.

Then, we took unlabeled data and predicted the output over the cloud. The task was very simple thanks to the API simplicity of IBM-PyWren.

<https://github.com/facebookresearch/fastText>

## Comparing local prediction vs. IBM cloud function prediction

AG news

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Data size (input)** | **Chunk Size** | **Map instances** | **Local time to complete [sec]** | **Serverless using PyWren time to complete**  **[sec]** |
| **Test 1** | 29MB | 4MB | 7 |  |  |
| **Test 2** |  |  |
| **Test 3** |  |  |
| **Test 4** |  |  |
| **Test 5** |  |  |
| **Average** | 29MB | 4MB | 7 |  |  |

## 

DBpedia

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Data size (input)** | **Chunk Size** | **Map instances** | **Local time to complete [sec]** | **Serverless using PyWren time to complete**  **[sec]** |
| **Test 1** | 172.8MB | 6MB | 28 |  |  |
| **Test 2** |  |  |
| **Test 3** |  |  |
| **Test 4** |  |  |
| **Test 5** |  |  |
| **Average** | 172.8MB | 6MB | 28 |  |  |

Yelp reviews

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Data size (input)** | **Chunk Size** | **Map instances** | **Local time to complete [sec]** | **Serverless using PyWren time to complete**  **[sec]** |
| **Test 1** | 472.8MB | 10MB | 46 |  |  |
| **Test 2** |  |  |
| **Test 3** |  |  |
| **Test 4** |  |  |
| **Test 5** |  |  |
| **Average** | 472.8MB | 10MB |  |  |  |

Sogou news

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Data size (input)** | **Chunk Size** | **Map instances** | **Local time to complete [sec]** | **Serverless using PyWren time to complete**  **[sec]** |
| **Test 1** | 1.26GB | 16MB | 8 |  |  |
| **Test 2** |  |  |
| **Test 3** |  |  |
| **Test 4** |  |  |
| **Test 5** |  |  |
| **Average** | 1.26GB |  |  |  |  |

# Phase B

## Training and testing a fastText model

Next, we decided to show the IBM-PyWren capabilities regarding the hyperparameters problem. In order to do so, we decided to train and test the models over the cloud. We implemented a generic class that its input is a train and test functions that used K fold cross validation that we distributed over IBM cloud for better performance.

TODO graph and extra info here

## Comparing local training vs. IBM cloud functions training

AG news

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Data size (input)** | **Chunk Size** | **Map instances** | **Local time to complete [sec]** | **Serverless using PyWren time to complete**  **[sec]** |
| **Test 1** | 29MB | 4MB | 7 |  |  |
| **Test 2** |  |  |
| **Test 3** |  |  |
| **Test 4** |  |  |
| **Test 5** |  |  |
| **Average** | 29MB | 4MB | 7 |  |  |

## 

DBpedia

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Data size (input)** | **Chunk Size** | **Map instances** | **Local time to complete [sec]** | **Serverless using PyWren time to complete**  **[sec]** |
| **Test 1** | 172.8MB | 6MB | 28 |  |  |
| **Test 2** |  |  |
| **Test 3** |  |  |
| **Test 4** |  |  |
| **Test 5** |  |  |
| **Average** | 172.8MB | 6MB | 28 |  |  |

Yelp reviews

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Data size (input)** | **Chunk Size** | **Map instances** | **Local time to complete [sec]** | **Serverless using PyWren time to complete**  **[sec]** |
| **Test 1** | 472.8MB | 10MB | 46 |  |  |
| **Test 2** |  |  |
| **Test 3** |  |  |
| **Test 4** |  |  |
| **Test 5** |  |  |
| **Average** | 472.8MB | 10MB |  |  |  |

Sogou news

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Data size (input)** | **Chunk Size** | **Map instances** | **Local time to complete [sec]** | **Serverless using PyWren time to complete**  **[sec]** |
| **Test 1** | 1.26GB | 12MB | 8 |  |  |
| **Test 2** |  |  |
| **Test 3** |  |  |
| **Test 4** |  |  |
| **Test 5** |  |  |
| **Average** | 1.26GB |  |  |  |  |

# Conclusions

Blab bla

# References

URLs blabla