Text Mining Project

"Programming Using NLP"

By Nadim Majed & Mehdi Ben Amor

Supervised by Dr. André Freitas

Table of Contents

1

Introduction

Introduction

1. Systems nowadays are becoming more and more AI based in terms of data processing,

2. One of the most popular tools to make that happen is Natural Language Processing know as NLP,

Motivation

1. **Not everyone wants to learn a programming language and code !?**

2. **So why don’t we offer the possibility of programming and coding by only using the natural language ?**

2

Proposed Model & Architecture

Proposed Model & Architecture (1/7)

Our Model in short:

- **Query processing (Syntaxic tree parsing and POS tagging)**
- **Features Extracting using Semantic Role Labeling.**
- **Semantic matching (WordVectorization, NER tagging, Formatting).**

Pipeline Architecture(2/7)

Simple timeline

Query Processing

- POS tagging
- Syntaxic Tree

Feature extraction

- Semantic Role Labeling

Semantic Matching(I): Method guessing

- Word2Vec trained Model
- Similarity

Semantic Matching(II)

- Word2Vec
- NER tagging
- Formatting

Execution

- Code executing

Pipeline Architecture(3/7)

Query Processing:

- **Stanford CoreNLP Annotator (“pos”, “parse”).**
- **Using the syntaxic tree for the next steps.**

Pipeline Architecture(2/7)

Simple timeline

Query Processing

- POS tagging
- Syntaxic Tree

Feature extraction

- Semantic Role Labeling

Semantic Matching(I): Method guessing

- Word2Vec trained Model
- Similarity

Semantic Matching(II)

- Word2Vec
- NER tagging
- Formatting

Execution

- Code executing

Pipeline Architecture(4/7)

Feature Extraction:

- **Semantic Role Labeling :**

  - Verbs(VB) as possible function name (word-level)

  - Noun Phrase(NP) as possible parameters (phrase-level)

- **List of possible parameters and function names .**

Pipeline Architecture(2/7)

Simple timeline

Query Processing

- POS tagging
- Syntaxic Tree

Feature extraction

- Semantic Role Labeling

Semantic Matching(I): Method guessing

- Word2Vec trained Model
- Similarity

Semantic Matching(II)

- Word2Vec
- NER tagging
- Formatting

Execution

- Code executing

Pipeline Architecture(5/7)

Method Guessing (SM-I):

- Using a pre-trained Word2Vec Model (INDRA server) to determine the method requested by the query.
- Computing similarity scores for all APIs in our dataset « actionKB ».
- Method guessing depends on :

  - Distance metric : « COSINE » or « EUCLIDEAN »

  - Precision variable: « ε » EPSILON

  - Similarity Score

Pipeline Architecture(2/7)

Simple timeline

Query Processing

- POS tagging
- Syntaxic Tree

Feature extraction

- Semantic Role Labeling

Semantic Matching(I): Method guessing

- Word2Vec trained Model
- Similarity

Semantic Matching(II)

- Word2Vec
- NER tagging
- Formatting

Execution

- Code executing

Pipeline Architecture(6/7)

Semantic Matching(II):Parameters

- **Parameters matching process is on three parts:**

  - « NER tagging » : if « possible ».

  - « Similarity score » between possible parameters and the method's parameter.

- **Third part is « Formatting » (used for checking/selecting) :**

  - Number

  - Independent sentence/word: entered between « " " » in the query.

  - Specific Format : deduced from a prepared dataset.

Pipeline Architecture(2/7)

Simple timeline

Query Processing

- POS tagging
- Syntaxic Tree

Feature extraction

- Semantic Role Labeling

Semantic Matching(I): Method guessing

- Word2Vec trained Model
- Similarity

Semantic Matching(II)

- Word2Vec
- NER tagging
- Formatting

Execution

- Code executing

Pipeline Architecture(7/7)

Execution

- **Executing the instantiated code of the guessed methods**

Live DEMO

Programming
NLP

3

Evaluation

Evaluation (1/4)

Defining the variables

EPSILON (ε)

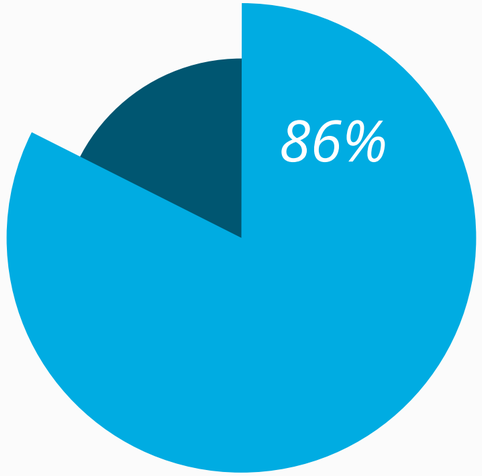
- **Tolerated Error in method matching scores**
- **[0.1, 0.08, 0.05]**

Similarity Metric

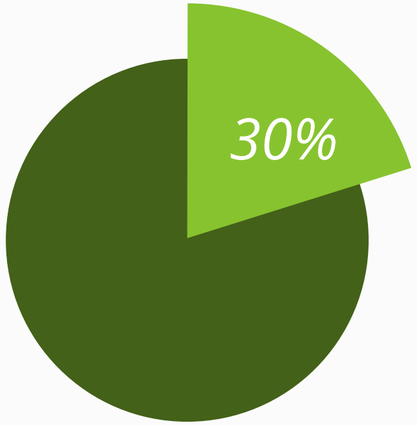
- **The score Function used to calculate the distance between two words**
- **['COSINE','EUCLIDEAN']**

Evaluation (2/4)

Performance Accuracy Graph



COSINE

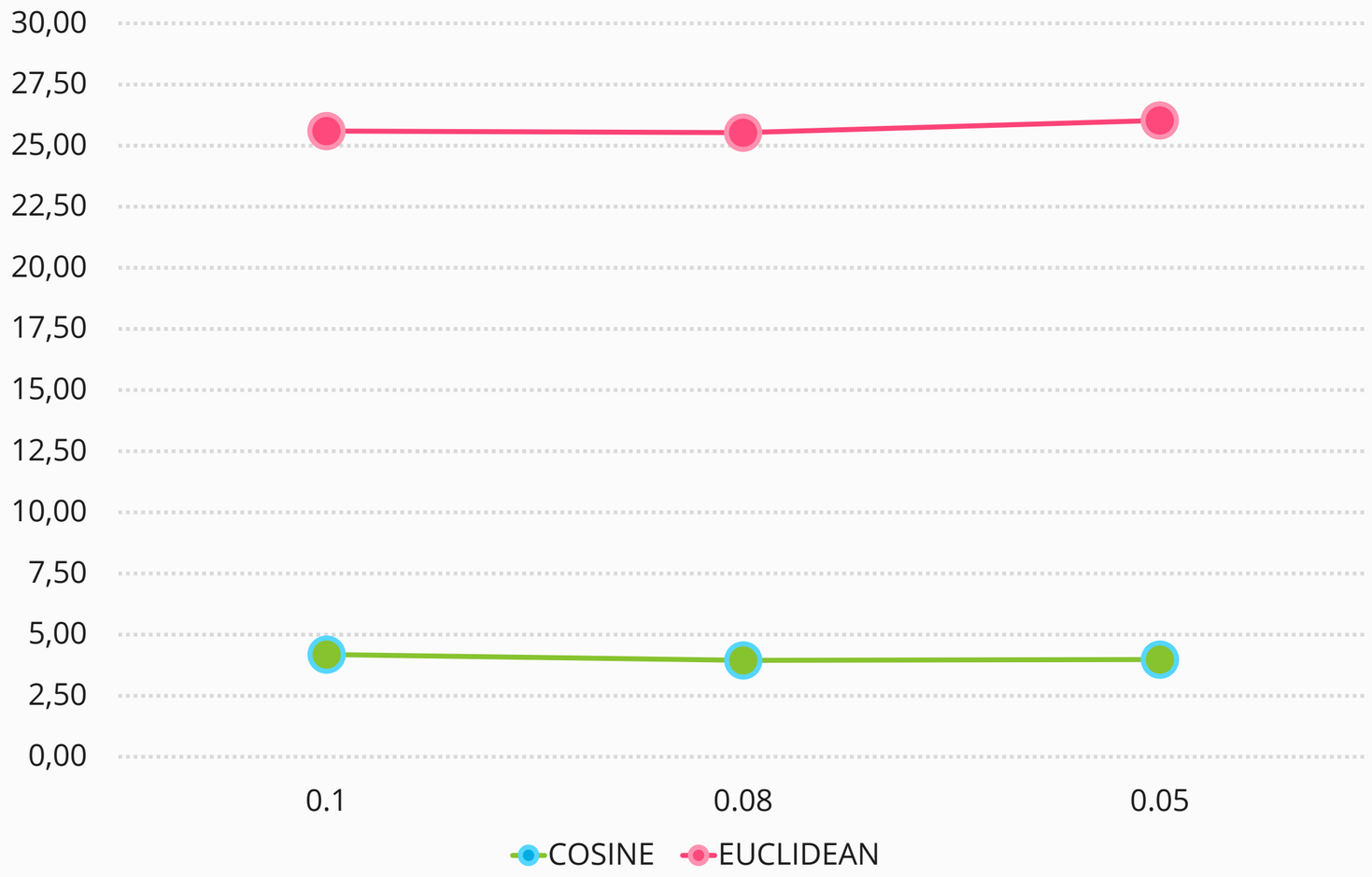


EUCLIDEAN

-The evaluation was based on 30 queries covering our 25 API methods . (only 26 gave the right code)

Evaluation (3/4)

Average Execution time for each query processing



Evaluation (4/4)

Optimization:

- **Most optimal choice of variables values will be:**

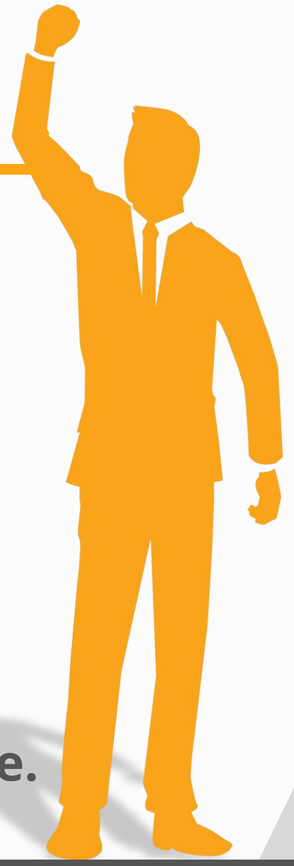
  - EPSILON=0,08

  - Metric=COSINE

4

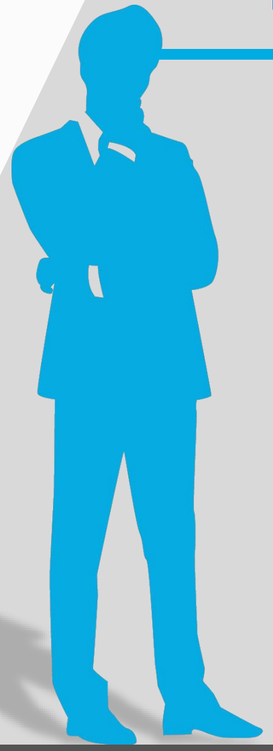
Post-Mortem Analysis

Worked!



- The query contains necessary parameters.
- The signature of the api is well structured.
- The name of the function is meaningfull and distinctable.

Failed:¬(



- Similarity between the names of the methods.
- Number of parameters is lower than other functions.
- Vague function name: get, search, ...

Post-Mortem Analysis

4

Conclusion

Conclusion

Important points

1. We have used multiple text mining technics.
2. Linguistic structure depends on every person in the world that's why it's hard to get a high accuracy.
3. Keep looking for optimizing all the time

Future Goals

"User Friendly" Interface

Creating an interface for an easier use.

Model Training

Introducing the machine learning approach to train our model .

APIs' methods DataSet

Enlarge our methods dataset to become more diversified and wider.

Thank You for Listening

...

Any Questions?