**Information Retrieval – Dr Inbal Budowski Tal - IDC**

**HW4 – KNN Classification Report**

This project implements the KNN algorithm for document classification. It is comprised of a training set which is given as an initial input the classifier and a test set for profiling the accuracy of the classifier.

This report will cover the main modules of the system, the design and functional choices we made implementing it and the benchmark tests we have conducted in different milestones of the project.

Table of contents:

1 Into

2 System Design

3-4 Milestones in the project and test results

**System Design**

The system is composed of 4 main components:

* **Config –** responsible of parsing parameters file.
* **CsvParser –** parses input files (training and test sets).
* **IndexingEngine –**responsible for indexing the training set and supply metadata for computing tf-idf values.
* **SearchEngine –** responsible for computing tf-idf vectors and neighbors of test documents based on similarity to indexed training set.
* **KNNClassifier –** classifies test document based on an initialization with training data.
* **Label –** represents a measurement over a single label

**The flow of the system is done in two phases:**

Phase 1 – initialization: Phase 2 – Classification:





**Milestones in the project and test results**

Designing and implementing this project has been challenging in few aspects: Scalability, Efficiency and design.

First, we started implementing the KNN Classification based on the search engine we have implemented in the previous task, using its similarity method for comparing each document as a query to find the best k results in the corpus (our dataset).

This required us to redesign our system to fit our new demands, especially in resource consuming aspect since indexing and searching through the huge corpus took a lot of time and CPU and RAM resource.

After rewriting some of our inefficient code, we managed to get the classification working for significantly more documents – mostly using code analyses tools.

Secondly, we wanted to benchmark our system and find the optimal k for our use – running our algorithm for every k in the range of 1 up to 100, following are the results we have found:

Lastly, we started thinking about ways to make our results better – starting with implementing title indexing and searching, understanding that the title has maybe the most important words to index (since most of the time it contains the key words of the document).