Introduction

Chapter 1

Conception

In this section we present the design of the solution built during the internship with its different components. The main focus was to find a relevant metric (from **??**) to measure the similarity between two columns, and it is the angular metric that has been used as it takes into account the magnitude of each point and it responds well to the problem of big dimensions presented by the euclidean metric, also, it is not limited to measure similarities over discrete data as the Jaccard similarity does.

The choice of the angular metric implies that the LSH method for the indexation process has to be chosen from the ones that estimate this metric. We go for cross-polytope as LSH method to estimate the similarities of the euclidean metric on a unit sphere (which has the same order of magnitude as the angular metric).

Our solution is divided into two main parts, the first part is related to the indexation process where the dataset indexes are computed and stored, and the second part is meant to take the process of detecting similarities between datasets through their previously computed indexes.

1.1 Indexation

As described previously, this phase consists of computing the index of each dataset's column and to store it in a database. This process is completed through the following three steps:

1.1.1 Preprocessing

The indexes are based on the column types, 2 types are considered: textual columns and non textual columns (numerical columns). The preprocessing will be different in function of the column separation process:

Non-Textual (Numerical) columns

One of cross-polytope requirements is that the data should have a size equal to a power of 2, for this we'll set this size on a value between 128 to 2048, and we'll call it DEFAULT_DATA_SIZE.

A random sample of a size DEFAULT_DATA_SIZE is picked for each column and then normalized to have the data ready to be indexed with cross-polytope.

Textual columns

The textual columns have a different type of preprocessing, is it based on embedding them with a pre-trained embedding model according to a certain approach that indicates how the embedding model is used over all the cells to have at the end on vector representing the column content. Four approaches will be tested to pick the most accurate one.

Approach 1 : The idea of this approach is to concatenate the texts of each cell to have one text that will be embedded with the chosen embedding model. The output will depend on this embedding model, and the last step will be to pad with zeros this output to make it of a size equal to a power of 2.

Approach 2 : This approach consists of embedding each cell on its own with the chosen embedding model, and then to concatenate this embeddings to form one vector that will be the output of this approach.

Approach 3 : This approach has the same logic of the previous one, it adds an extra step which is to perform a PCA on each embedding just before concatenating in order to have more control on the size of the output and to make it easily satisfy the cross-polytope conditions.

Approach 4 : This approach has also the same logic of the second one, meaning that it starts with embedding each cell independently, and then get the mean of each component of the embedding vector.

$$a_{i/i\in[1,m]} = \frac{1}{n}\sum_{j=0}^{n} e_{j,i} \tag{1.1}$$

Where:

- m is the size of the embedding model's output.
- n is the size of the column.
- e_j is the embedding of the j^{th} cell.
- $e_{j,i}$ is the i^{th} component of the j^{th} cell embedding.

The last step in this approach will be to pad with zeros this output to make it of a size equal to a power of 2.

This step takes as input the concerned dataset, and outputs a list of columns with their preprocessed data for both textual and non-textual data.

1.1.2 Computing index

At the level of computing indexes, we get the preprocessed column groups and we pass both of them to the indexation phase by applying Multi-Probe Cross-Polytope LSH.

Two parameters need to be tuned at this level, the number of hashing functions to apply by Cross-Polytope LSH, and the number of probes to consider by Multi-Probe. The choice of these two parameters is explained in the section 2

This step outputs two group of indexes, the group of textual column indexes, and the group of non-textual column indexes. Each column index is represented by an array of 2 dimensions, where each row represents the result of the corresponding hashing function,

and each column represent an element of the probing sequence for this hashing function result.

The figure 1.1 illustrates the process of preprocessing and computing the indexes of a dataset.

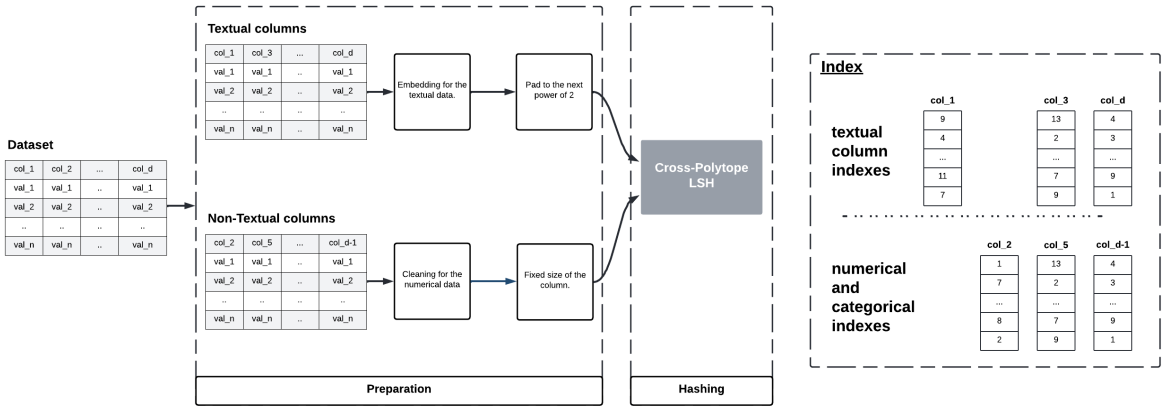

Figure 1.1: Indexation process illustration

1.1.3 Storing index

The indexes computed in the previous step need to be stored in order to be used lately in similarity detection. A key is assigned to every dataset in Talend Data Inventory, this key is used to store the indexes corresponding to each dataset. The indexes are stored in a dedicated database which contains the following attributes:

- **dataset_id**: Integer. The ID of the corresponding dataset.
- **column_name**: String. The name of the index column.
- **column_type**: String. The type of the index column.
- **signature**: Bytes. The encoding of the index.

1.2 Similarity Detection

This second part of our solution is meant to proceed for the similiarity detection between two dataset indexes. As described previously, we scale the similarity detection between dataset to the similarity detection between their columns, and from the column similarities we can give a score representing the dataset similarities.

The process of detecting similarities between columns is done by the following steps:

- Use two groups of bucket, the first one for the textual columns and the second for the non-textual column.
- Assign each column to a number of buckets in function of its calculated index and its type.

- Measure the collision rate between the columns that share at least one bucket to give their similarity measure (a score between 0 and 1).

The similarity measure given by this method is inversely related to the angular distance, meaning that a score close to 1 represents a high similarity, and a score close to 0 represents a low similarity.

1.3 Implementation

To implement the solution explained in the two previous sections, five different components have been used:

- **Preprocessing**: It consists of the classes responisble for the preprocessing phase, its main class is *DataPreprocessing* that assures the different types of preprocessing (noramlization, sampling, embedding..) with the use of two other classes, *Embedder* and *EmbeddingModel* used for embed the textual columns of a dataset.
- **Hashing**: It is the component that assures the indexation process, two classes have been used for computing the indexes: *CrossPolytope* and *MultiProbeCrossPolytope*, while the class *Index* is used to represent the index computed for a dataset,
- **Storing**: It consists of one class that manages the storing of the computed indexes
- **Query**: This component gives the features of detecting the similarities between indexes through the class *LSHQuery*, and to manage their results through the class *QueryResult*
- **Plotting**: With this components, the results given by *QueryResult* are represented graphically with different types of diagrams.

The figure 1.2 represent the class diagram with its diffrents components as explained in the previous paragraph.

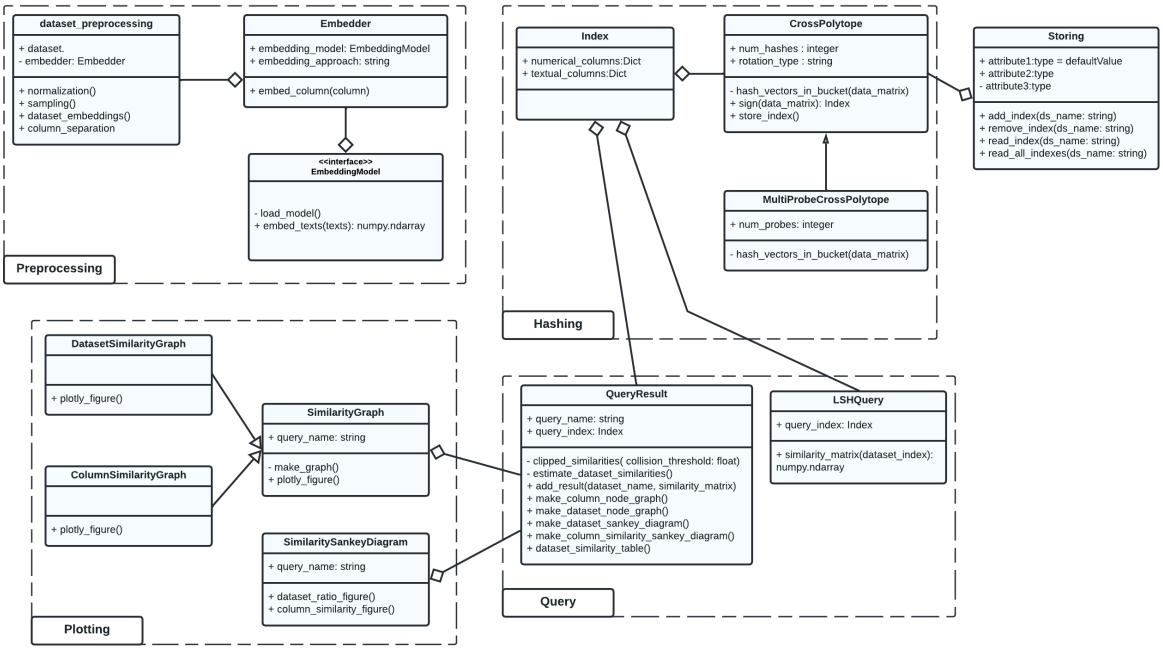

Figure 1.2: Class Diagram

Chapter 2

Experiments and results

2.1 Test and Experiments

In this section, the focus is on finding the combination of hyper-parameters and embedding approach leading to the best similarity estimation. To do this we do our experiments under two metrics.

- **Embedding approach metric**: It consists of applying the different embedding approaches proposed in (mention the section when it's done) to get the textual column representations of each dataset couple, and then measuring the similarities between these representations to compare them with the ground truth labels 2.1. The similarity are measured with the theoretical collision probability that we estimate by cross-polytope LSH:

$$P(h(x) = h(y)) = \exp -\frac{\pi^2}{4 - \pi^2}. \ln d \tag{2.1}$$

  .

- **Cross-Polytope hyper-parameters:** It consists of estimating the similarities with cross-polytope using different values for the two parameters: number of hashing functions to use, and the number of probes, and computing the error of this estimation to the theoretical collision probability (2.1).

The set of labelled similarities : It is a dataset containing different sets of datasets associated with lists of columns that can be matched. It is a human labelling done in a previous project at Talend. This labelled datasets are organized as the following example:

dataset 1	**dataset 2**	**column dataset 1**	**column dataset 2**
students	clients	Student Name	client name
students	clients	inscription data	join date
students	clients	Gender	gender
books	publications	release data	publication date
.	.	.	.
books	publications	title	title

To each couple of labelled dataset with their respective similar columns, a label matrix is associated to represent their similarities with the following process:

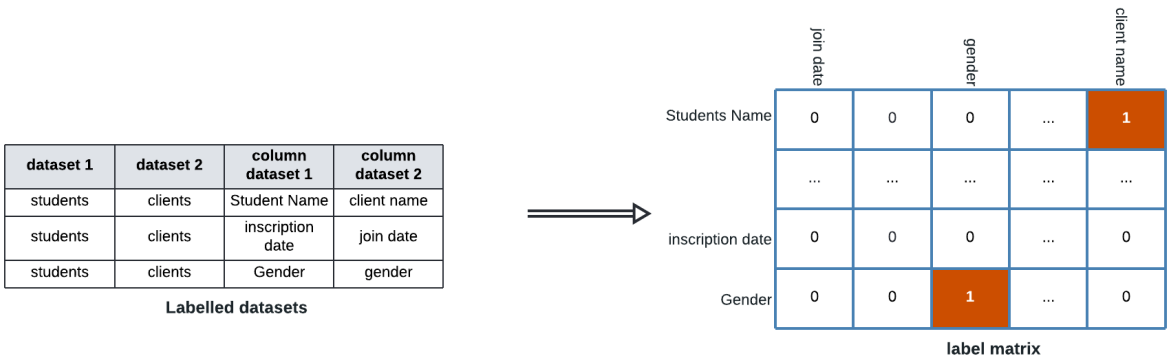

Figure 2.1: Label matrix generation from the labelled column similarities

2.1.1 Embedding approach metric

In this section, the goal is to measure at each time the accuracy of detected similarities compared to the validated ones (in the labelled similarities dataset), with changing the embedding model and the embedding approach.

The embedding models:

- Universal Sentence Encoder Multilingual (USEM)
- Paraphrase Multilingual
- Neural-Net Language Models (NNLM)
- Sentence transformer: distiluse-base-multilingual

The embedding approaches:

- **Approach 1**: (or Concatenate and embed) Concatenate the texts and do the embeddings
- **Approach 2**: (or Concatenate embeddings) Perform the embedding per cell and concatenate the results.
- **Approach 3**: (or PCA on embeddings) Perform the embedding per cell, reduce their dimension with PCA and concatenate.
- **Approach 4**: (or Mean on embeddings) Perform the embedding per cell, and get the mean for each component in the result embedding vectors.

To compute this accuracy, we represent the estimated similarities between couples of datasets in a form of a matrix where each axis represent dataset columns. The next step is to compute for each dataset couple the accuracy of the estimated similarities between the columns that has been labelled similar.

To define this metric mathematically, let E be the estimation matrix, and L be the labelled similarity matrix, and 1_L the number of occurrences of 1 in L

$$metric = \frac{1}{1_L} \sum_i \sum_j L_{i,j} * E_{i,j} \tag{2.2}$$

The figure 2.2 shows an example of two datasets with 4 and 5 columns respectively.

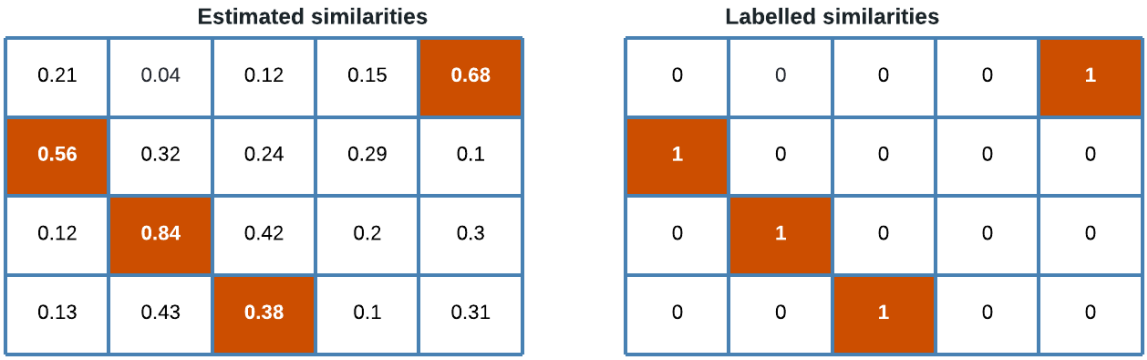

Figure 2.2: Embedding metric computing example

The metric in this case is calculated by picking up the similarities corresponding to labelled similarities and computing their means:

$$metric = \frac{1}{4}(0.68 + 0.56 + 0.84 + 0.38) = 0.615 \tag{2.3}$$

The table 2.1 presents the results of this test with every combination of embedding model with an embedding approach.

	USEM	**NNLM**	**Paraphrase Multilingual**	**Distiluse**
Concatenate and embed	0.301	0.580	0.409	0.365
Concatenate embeddings	0.044	0.06	0.085	0.107
PCA on embeddings	0.275	0.319	0.403	0.350
Mean on embeddings	0.494	0.583	0.572	0.554

Table 2.1: Results for the embedding choice metric

Some remarks can be done over this results:

- The best accuracies were around $0.56, 0.58$.
- The fourth approach gave the best results for each embedding model.
- The notion of angular distance loses its sense when the dimensions are too big case of approach 2 (Concatenate embeddings). As each cell with this approach will be in a dimension of 512 or 368 (depending on the embedding model used), the vector of concatenated word representations will become too large, which will make the angular metric loses some relevance, and also the concatenation of the embeddings won't give a good comprehension of the column content.
- The accuracies depend on the embedding model understanding of different forms of texts, as there may be some forms that are not understandable, for example: links, email addresses, acronyms, ect..

The configuration selected after this experiment is:

- **Embedding model:** Paraphrase Multilingual
- **Embedding approach:** Approach 4; perform the embedding per cell, and get the mean for each component in the result embedding vectors.

2.1.2 Indexation hyper-parameters

In this second test the goal is to find the best combination of number of the hashing functions to use in indexation and the number of probes to use in Multi-Probe LSH. The best combination is the one that gives the index with the minimum error of estimation of the angular distance. To do this, a list of values for each of the two variables and a set of 1000 dataset couples have been set, and the error for a combination of a number of probes and a number of hashing functions is computed by computing the mean absolute error for each couple of datasets and computing the mean of error over all the list of couples.

The pseudo-algorithm 1 describes the steps to complete this test routine.

Algorithm 1: Pseudo-algorithm for the similarity estimation test routine.

```
Input: num_probes: List of integer;
num_hashes: List of integer;
dataset_list: List of dataset couples.
error_matrix: matrix
for (num_probe, num_hash) in (num_probes, num_hashes) do
    errors = []
    for (dataset_1, dataset_2) in dataset_list do
        error = similarity_estimation_error(dataset_1, dataset_2) add_element(errors,
         error)
    end
    error_matrix[num_probe, num_hash] = mean(errors)
end
return error_matrix
```

The similarity_estimation_error function is computed by indexing the datasets in input with the corresponding configuration and compute the mean absolute error with the exact similarities.

The results of this test are represented in the following table where the columns correspond to the number of probe values and the rows correspond to the number of hashing functions.

The table 2.2 shows the results of this experiment, from which we can take the following points:

- The best configurations gave errors in the interval of $[0.45, 0.55]$.
- The results confirms the theoretical result in (Andoni et al., 2015) saying that by using a multi-probe technic we can improve the results of estimation without having to increase the number of hashing functions used, for example, the difference of estimation error between the case of 256 hashing functions and 1024 hashing functions is at the range of 0.003.

Number of Probes / Number of hashing func	**3**	**4**	**5**	**6**	**7**	**8**
16	0.084150	0.091189	0.102268	0.114099	0.126253	0.138592
32	0.069658	0.073694	0.084380	0.097009	0.110436	0.123476
64	0.060197	0.063477	0.075861	0.090637	0.105167	0.119083
128	0.054985	0.056040	0.069854	0.086120	0.101522	0.115687
256	0.049240	0.055169	0.072686	0.089780	0.105259	0.119369
512	0.047183	0.053220	0.072063	0.089319	0.104874	0.119079
1024	0.046043	0.052195	0.071337	0.088481	0.103913	0.117931

Table 2.2: Results for the multi-probe hyper-parameters testing

Bibliography

Andoni, A., Indyk, P., Laarhoven, T., Razenshteyn, I., and Schmidt, L. (2015). Practical and optimal lsh for angular distance.