1. Building the Wikipedia Hierarchy

Our goals:

- Download the Wikipedia hierarchy data (pages, links between them and categories).
- Build a graph based on Wikipedia's pages and the connections between them.
- Build a tree from the categories of Wikipedia's pages.
- Define groups on the graph, each group made from nodes (pages) that are connected to the same category from the category tree.
- Train and build a model to project the groups on the graph and create a metric in a way that reflects the connection between groups that are closer in the tree. For example, 2 groups under the same category will get higher score than 2 other groups that are under foreign categories.

Download the data:

From Wikipedia dumps, at https://dumps.wikimedia.org/enwiki/latest/ - we downloaded the files:

```
enwiki-latest-category.sql.gz        //size after unzipping: 0.09 GB
enwiki-latest-categorylinks.sql.gz   // 19 GB
enwiki-latest-page.sql.gz            // 5GB
enwiki-latest-pagelinks.sql.gz       //46.8 GB
```

unzip (with gunzip) and save the files.

Open a database using mysql:

Mysql -uroot -ppassword

```
> CREATE DATABASE WIKI;
```

```
> USE WIKI;
```

Now exit mysql, and from the command line, run the following:

Mysql -uroot -ppassword WIKI < `enwiki-latest-category.sql`

And so for each of the files.

NOTE: These commands may take long (depends on your cpu's, but it may take couple of days), therefore it's recommended to run it with nohup &.

Explanation about the data, Creating the category tree and the graph:

Python scripts for creating the tree and graph can be found here:

https://github.com/rachelic44/Wikipedia-Hierarchy

These 4 tables created from the files, cover the graph and the tree's nodes and edges. Here is some explanation about the tables:

```
MariaDB [WIKI]> show columns from page
    -> ;
+--------------------+---------------------+------+-----+---------+----------------+
| Field              | Type                | Null | Key | Default | Extra          |
+--------------------+---------------------+------+-----+---------+----------------+
| page_id            | int(8) unsigned     | NO   | PRI | NULL    | auto_increment |
| page_namespace     | int(11)             | NO   | MUL | 0       |                |
| page_title         | varbinary(255)      | NO   |     |         |                |
| page_restrictions  | tinyblob            | NO   |     | NULL    |                |
| page_is_redirect   | tinyint(1) unsigned | NO   | MUL | 0       |                |
| page_is_new        | tinyint(1) unsigned | NO   |     | 0       |                |
| page_random        | double unsigned     | NO   | MUL | 0       |                |
| page_touched       | varbinary(14)       | NO   |     |         |                |
| page_links_updated | varbinary(14)       | YES  |     | NULL    |                |
| page_latest        | int(8) unsigned     | NO   |     | 0       |                |
| page_len           | int(8) unsigned     | NO   | MUL | 0       |                |
| page_content_model | varbinary(32)       | YES  |     | NULL    |                |
| page_lang          | varbinary(35)       | YES  |     | NULL    |                |
+--------------------+---------------------+------+-----+---------+----------------+
13 rows in set (0.00 sec)
```

Table 1.1 : Page

```
MariaDB [WIKI]> show columns from pagelinks;
+--------------------+-----------------+------+-----+---------+-------+
| Field              | Type            | Null | Key | Default | Extra |
+--------------------+-----------------+------+-----+---------+-------+
| pl_from            | int(8) unsigned | NO   | PRI | 0       |       |
| pl_namespace       | int(11)         | NO   | PRI | 0       |       |
| pl_title           | varbinary(255)  | NO   | PRI |         |       |
| pl_from_namespace  | int(11)         | NO   | MUL | 0       |       |
+--------------------+-----------------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```

Table 1.2: Pagelinks

```
+-----------------+--------------------------+-----+-----+-------------------+--------------------------+
| Field           | Type                     | Null | Key | Default           | Extra                    |
+-----------------+--------------------------+-----+-----+-------------------+--------------------------+
| cl_from         | int(8) unsigned          | NO   | PRI | 0                 |                          |
| cl_to           | varbinary(255)           | NO   | PRI |                   |                          |
| cl_sortkey      | varbinary(230)           | NO   |     |                   |                          |
| cl_timestamp    | timestamp                | NO   |     | CURRENT_TIMESTAMP | on update CURRENT_TIMESTAMP |
| cl_sortkey_prefix | varbinary(255)         | NO   |     |                   |                          |
| cl_collation    | varbinary(32)            | NO   | MUL |                   |                          |
| cl_type         | enum('page','subcat','file') | NO   |     | page              |                          |
+-----------------+--------------------------+-----+-----+-------------------+--------------------------+
7 rows in set (0.00 sec)

MariaDB [WIKI]>
```

Table 1.3: categorylinks

```
MariaDB [WIKI]> show columns from category;
+-------------+------------------+------+-----+---------+----------------+
| Field       | Type             | Null | Key | Default | Extra          |
+-------------+------------------+------+-----+---------+----------------+
| cat_id      | int(10) unsigned | NO   | PRI | NULL    | auto_increment |
| cat_title   | varbinary(255)   | NO   | UNI |         |                |
| cat_pages   | int(11)          | NO   | MUL | 0       |                |
| cat_subcats | int(11)          | NO   |     | 0       |                |
| cat_files   | int(11)          | NO   |     | 0       |                |
+-------------+------------------+------+-----+---------+----------------+
5 rows in set (0.01 sec)
```

Table 1.4: category

- **The page** table contains 48 Million records, which are the graph nodes.
  Each node (page) has the page_namespace attribute. All pages who represent categories will be defined by namespace 14. All regular pages are defined with namespace0 , files are defined with namespace 6, and so on.
- **The page links** table contains 1.1 Billion records, which are the edges in the graph – each line in this table represents an edge from pl_to to pl_from .

  NOTE: pl_from is the name of the target node. In order to find its page number = the node, run the following from mysql command: SELECT page_id from page where page_title like ('pl_title') and page namespace = 'pl_namespace'.

- **The category** table contains 1.8M records, which are the nodes in the tree. These nodes can also be found in the page table with page_namespace = 14.
- **The categorylinks** table contains 135M records which define the connections between categories. A line in the table defines that cl_from is the son of cl_to in the category tree. Same NOTE as in the pagelinks, in order to find the page_id of the category of cl_to (which is the name of the category), Run: SELECT page_id from page where page_title like ('cl_to') and page_namespace = 14.

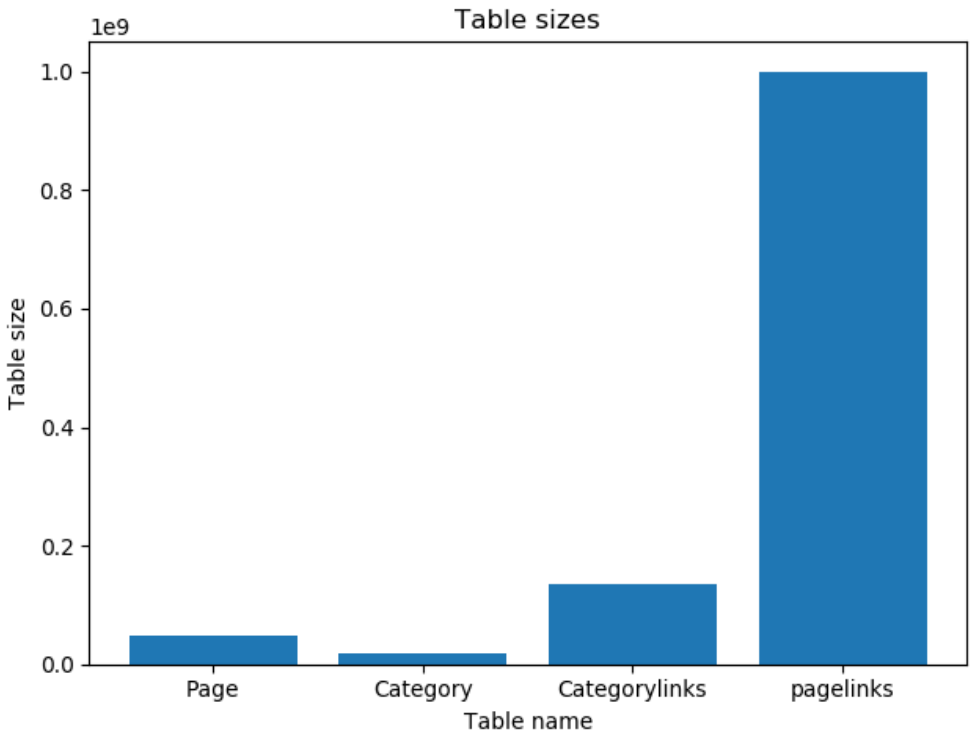

Figure 1.1: Table sizes

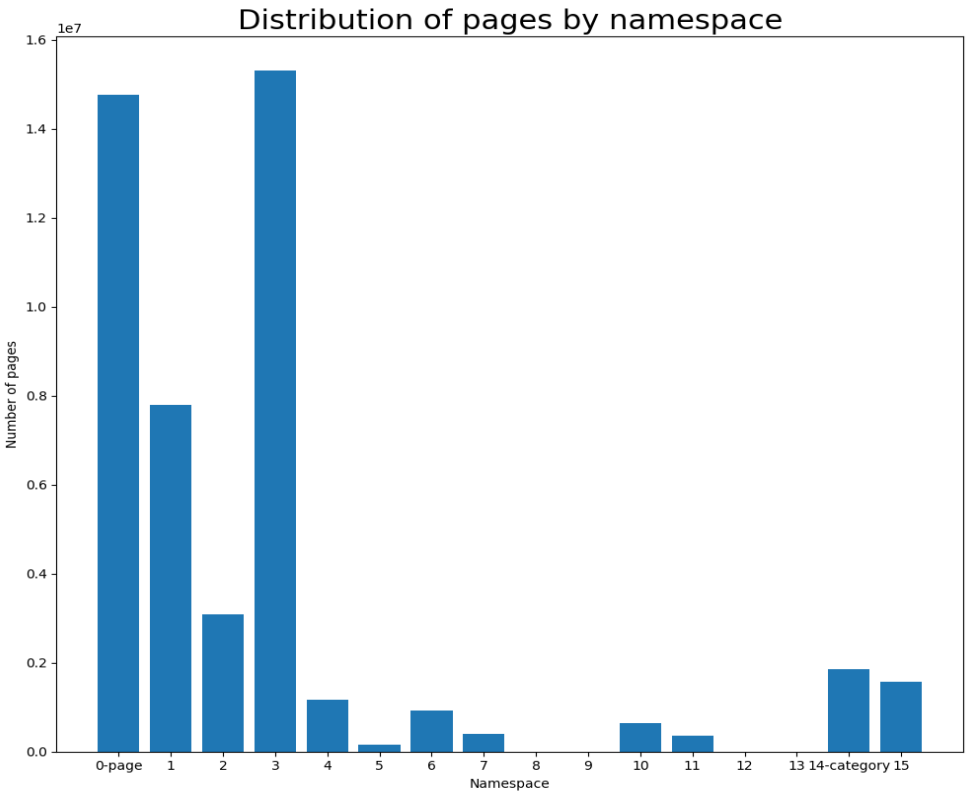

Figure 1.2: Distribution of pages by namespace

Define groups on the graph:

As first, we chose 4 supervised groups for which we knew the hierarchy (Fig 1.3, 1.4 for description of groups and sizes).

The python script for choosing the groups and creating the demo-graph can be found here:

https://github.com/rachelic44/Wikipedia-Hierarchy

For each node in each group, we computed all its first and second neighbors and created a large graph containing altogether 290,649 nodes and 1,589,345 edges. We projected the graphs over 7 dimensions using node2vec, and then took 10 samples of pairs of groups either within a community (i.e. two non-overlapping groups of nodes within the same community), or from different communities. We then applied different metrics over each pair of projections and different heuristics to compute the distance between groups, and tested whether within communities pairs of groups are close than between communities pairs of groups (Figure 1.5). The different metrics and heuristics are detailed Figure 1.5.

The python script for the metrics and heuristics can be found in the former link.

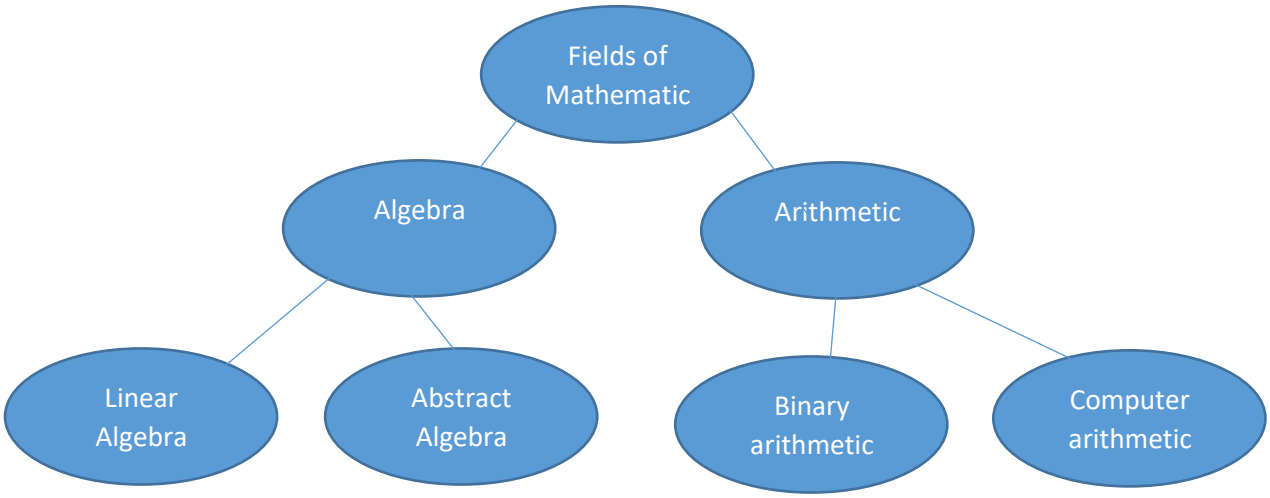

Figure 1.3: Hierarchy of groups used for the comparison

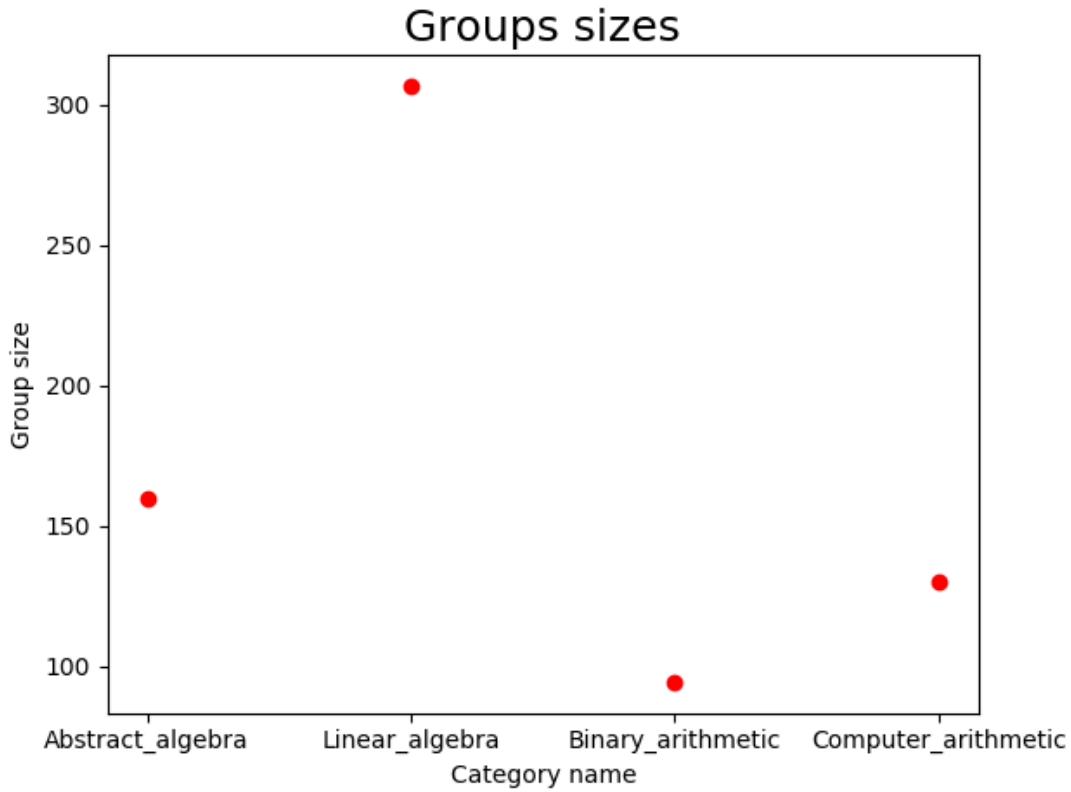

Figure 1.4: Sizes of the groups used for the comparison

Train the model:

Results for the 4 groups:

Using Anova score (f_oneway): each cell represents the (f_value, p_score) values, averaged over trials.

Vectors Metric / Groups metric	Min	Max	Avg	Centroid
euclidian	(5.58 ,0.013)	(4.37 ,0.029)	(11.94 ,0.0005)	(250.99 , 2.29e-13)
mahalanobis	(2.67 ,0.97)	(1.87 ,0.18)	(18.29 ,0.003)	(160.86 , 9.15e-12)
hamming	(11.73 ,0.006)	(2.67 ,0.5)	(7.27 ,0.005)	(70.86 , 9.15e-11)
cityblock	(4.05 ,0.03)	(8.89 ,0.022)	(5.3 ,0.05)	(90.97 , 8.35e-10)
sqeuclidean	(2.6 ,0.1)	(4.98 ,0.019)	(9.5 ,0.001)	(117.6 , 1.1e-10)
correlation	(14.16 ,0.0002)	(2.014 ,0.16)	(81.72 , 1.95e-9)	(418.14 , 3.35e-15)

Table 1.5: Results on 4 groups

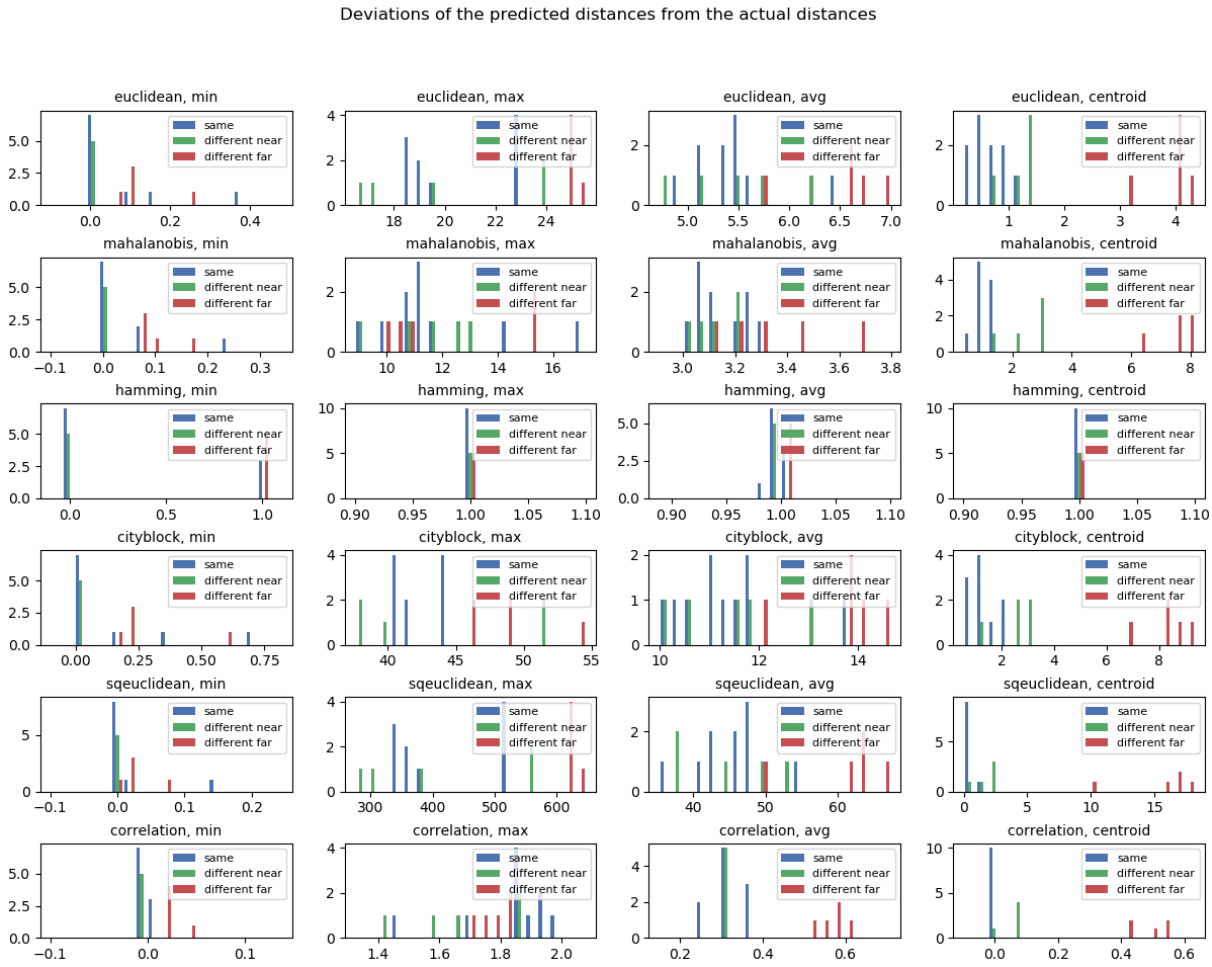

Figure 1.5: Results on 4 groups