Uniwersytet
Kardynała Stefana Wyszyńskiego
w Warszawie

Group Project – The languages proximity calculations

Anna Kelm
Mateusz Gwardys
Piotr Plebański
Filip Kocon
2022/2023

1. Project organization

A. Purpose of the project

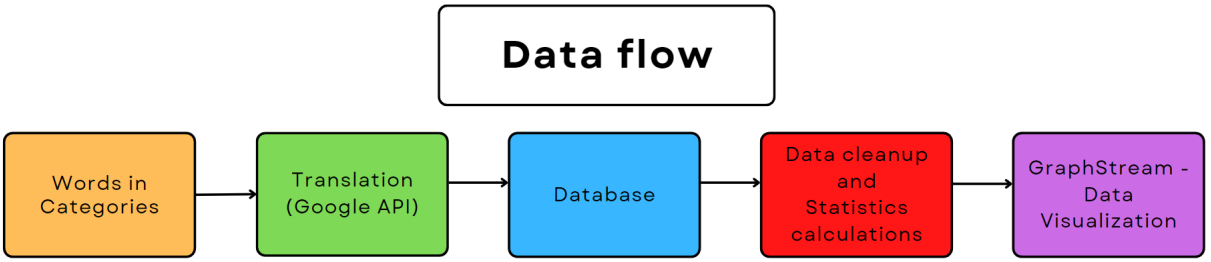
Purpose of the project is checking the similarity of languages based on statistical calculation and visualization of this data on the graphs.

B. Division of work

Our work was split into a couple of tasks that we could give to each member of the group.

- Words aggregation and translation - Matuesz
- Database setup - Mateusz
- Statistics calculation - Filip, Matuesz
- Graph visualization - Anna
- Maintaining repository - Piotr
- Project management - Anna, Piotr
- Report - Filip

Below is data flow for our processing inside our program.



C. Technology stack

- Python 3.10
- Java 1.8
- Maven

2. Language data and processing

A. Source of words and topics

Mateusz has found this data which source we officially cannot provide in this report, but if you will be interested we can provide it to you.

B. Topic list

- ANIMALS
- APPEARANCE
- ART
- BIOLOGY
- BIRDS
- BODY
- BUILDINGS
- BUSINESS
- CHANGE CAUSE AND EFFECT
- CLOTHES AND FASHION
- COLOURS AND SHAPES
- COMPUTERS
- COOKING AND EATING
- CRIME AND PUNISHMENT
- DANGER
- DIFFICULTY AND FAILURE
- DISABILITY
- DISCUSSION AND AGREEMENT
- DOUBT GUESSING AND CERTAINTY
- DRINKS
- EDUCATION
- ENGINEERING
- FAMILY AND RELATIONSHIPS
- FARMING
- FEELINGS
- FILM AND THEATRE
- FISH AND SHELLFISH
- FOOD
- GAMES AND TOYS
- GARDENS
- GEOGRAPHY
- HEALTHCARE
- HEALTH AND FITNESS

- HEALTH PROBLEMS
- HISTORY
- HOBBIES
- HOLIDAYS
- HOUSES AND HOMES
- INSECTS WORMS ETC
- JOBS
- LANGUAGE
- LAW AND JUSTICE
- LIFE STAGES
- LITERATURE AND WRITING
- MATHS AND MEASUREMENT
- MENTAL HEALTH
- MONEY
- MUSIC
- OPINION AND ARGUMENT
- PEOPLE IN SOCIETY
- PERMISSION AND OBLIGATION
- PERSONAL QUALITIES
- PHONES EMAIL AND THE INTERNET
- PHYSICS AND CHEMISTRY
- PLANTS AND TREES
- POLITICS
- PREFERENCES AND DECISIONS
- RELIGION AND FESTIVALS
- SCIENTIFIC RESEARCH
- SHOPPING
- SOCIAL ISSUES
- SPACE
- SPORTS BALL AND RACKET SPORTS
- SPORTS OTHER SPORTS
- SPORTS WATER SPORTS
- SUCCESS
- SUGGESTIONS AND ADVICE
- THE ENVIRONMENT
- TIME
- TRANSPORT BY AIR
- TRANSPORT BY BUS AND TRAIN
- TRANSPORT BY CAR OR LORRY
- TRANSPORT BY WATER

- TV RADIO AND NEWS
- WAR AND CONFLICT
- WEATHER
- WORKING LIFE

C. Languages list

- Celtic
  i. Welsh
- Germanic
  i. Danish
  ii. English
  iii. German
  iv. Norwegian
- Gaelic
  i. Irish
- Greek
  i. Greek
- Kartvelian
  i. Georgian
- Romance/Latin
  i. French
  ii. Italian
  iii. Latin
  iv. Portuguese
  v. Romansh
  vi. Spanish
- Slavic
  i. Bulgarian
  ii. Belarusian
  iii. Croatian
  iv. Czech
  v. Macedonian
  vi. Polish
  vii. Serbian
  viii. Slovak
- Uralic
  i. Estonian
  ii. Finish
  iii. Hungarian

D. Text translation and preprocessing

Google translate API was used for creating our word list, based on the English word list.

During this process we can end up with words that don't have their translation in google database, such as words from most slavic languages (see also: Evaluation Scores of Google Translate in 107 Languages). These words were left in their English version, which should be taken into account in the further similarity analysis.

After that we have used the "unidecode" library to transliterate words from non-latin alphabets and with diacritic marks, so that we can work with only A-Z text characters. Finally, we removed an insignificant number of words that turned out to be empty after translating and preprocessing steps.

E. Metrics and measures

A total of 5 metrics were used in the program, 4 of which provide distances between the words (Python *strsimpy* library), and one to evaluate the node proximity.

1. Levenshtein distance - the *strsimpy*, to the best our knowledge, uses the common understanding of Levenshtein distance, which is the minimal number character deletions, insertions or substitutions to transform one word to the other. [https://en.wikipedia.org/wiki/Levenshtein_distance]
2. Cosine distance - in cosine distance, each possible character from the character set corresponds to the dimension of a word vector. Word vector consists of counts of each unique letter in the word in their respective position. Cosine similarity *s(w1, w2)* is the dot product of two normalized word vectors. Cosine distance equals *1-s(w1, w2)*.
3. Jaccard distance – the Jaccard distance operates on the sets of unique letter in each word and is calculated as the ratio of cardinality of symmetric difference of the two word set to the cardinality of the intersection of the two sets.
4. Longest Common Subsequence (LCS) distance - the distance is calculated as the total sum of letters in each word after removal of the LCS.

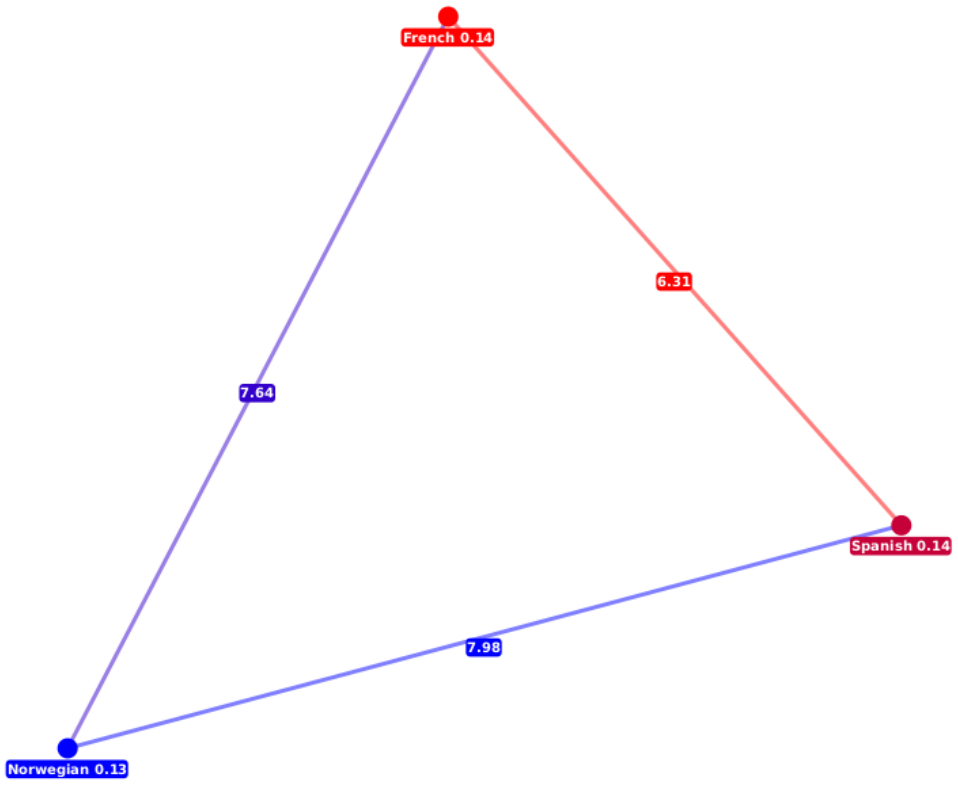
In order to indicate the relative position of the node within the graph, the node closeness centrality measure was used, defined as the reciprocity of the mean node distance to other nodes.

F. Data Visualization

The steps involved in calculating subsequent averaging the data to be displayed on the graphs are as follows:

- selection of the distance metric, the subset of languages, the subset of word categories and the object of the comparison – languages or categories – which will be displayed as the graph nodes,
- calculation of the mutual distances between the words in the selected languages; the words are limited by the categories selection,
- if the nodes are the languages, the edges lengths are the word distances averaged over the all words within each language pair,
- if the nodes are categories, the edge length is the mean distance between the words in each category jointly for all languages,
- the calculation of closeness according to the edge weights.

The graphs are visualized with the GraphStream library in Java. The figure below shows an example of the generated graph. Here, the graph nodes depict the Levenshtein distance between French, Spanish and Norwegian. The value on the edge label is the calculated distance; the edge color is blue for the higher distance values and red for the lower distance values. Nodes are labeled with the language and the value of the closeness centrality of the node in the presented graph. The red color of the node corresponds to its higher centrality in the graph.



3. Installation and user guide

This guide is also available in readme.me at the repository.

A. Setup

a. Prerequisites
   i. Python 3.10 (other versions not tested)
   ii. Java 1.8
   iii. Maven

b. Repository download

```
git clone https://github.com/akelm/languages_proximity_uksw.git
```

c. Build fat jar, create Python venv and install dependencies

```
chmod +x setup.sh
JAVA_HOME=<your-java-1.8-home-path> ./setup.sh
```

B. Execution

a. Generate graph from data, save and/or plot results

```
chmod +x run_graph_generator.sh
./run_graph_generator.sh

The script `run_graph_generator.sh` is a proxy to Python module `analyzeAndShow.graph_generator`,
which accepts the following arguments:

$ ./run_graph_generator.sh -h
usage: graph_generator.py [-h] --lang L [L ...] --cat C [C ...] [--node N] [--metric M] [--n_cpu N_CPU]
[--display] [--export]
Calculates differences between the selected languages over the selected categories and shows the results as
options:
 -h, --help            show this help message and exit
 --lang L [L ...], -l L [L ...] languages for comparison, at least 2
 --cat C [C ...], -c C [C ...] at least one category
 --node N, -n N        type of node displayed on the graph
 --metric M, -m M      a metric used for string distance calculations
 --n_cpu N_CPU, -p N_CPU number of cores used to compute distances
 --display, -d         should the graph be displayed in GraphStream
 --export, -e          should the graph be exported to PNG file in the current working directory
```

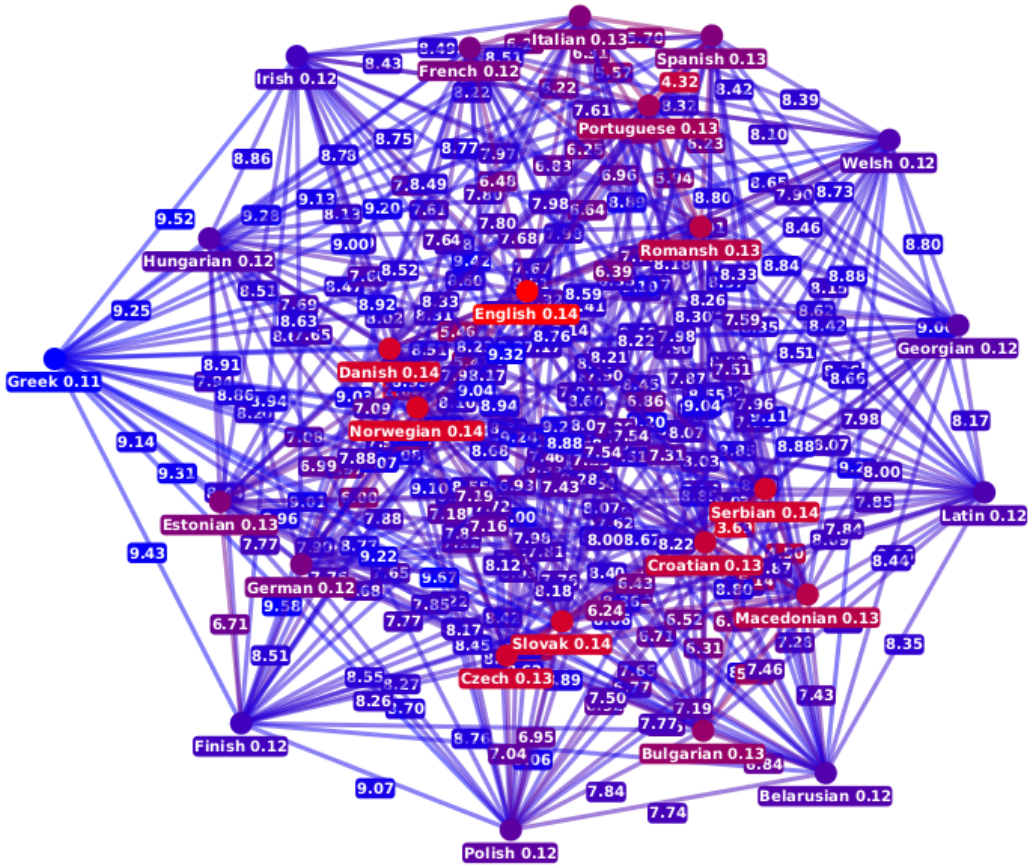
C. Disclaimer

Not tested on MacOS

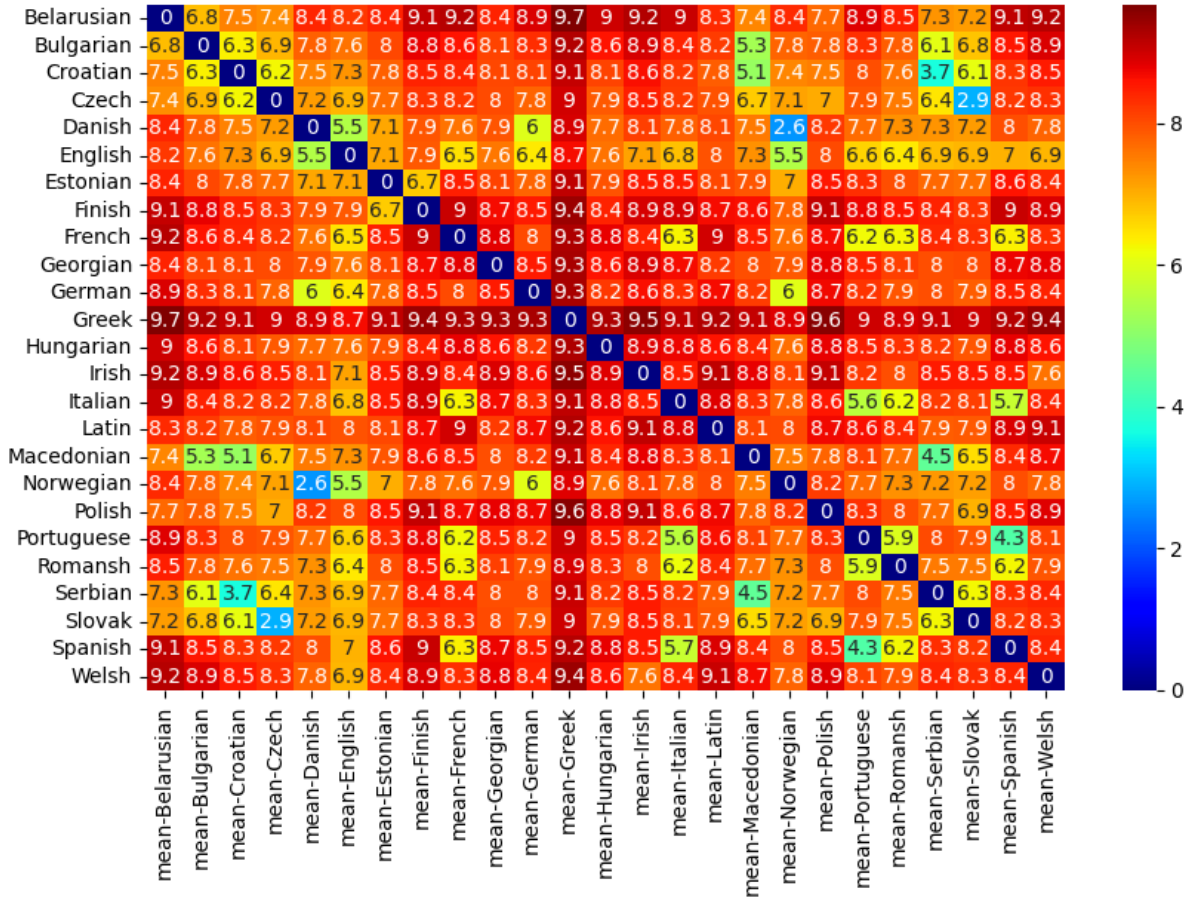
4. Results and discussion

We have found the deviations of the differences between the languages while comparing different word categories in particularly indicative of any cultural differences and, moreover, potentially highly influenced by the translation mistakes (specially in the slavic language family), therefore we analyzed the differences and similarities between the languages on the whole word set. The results provided below were calculated using Levensthein distance.

Although the graphs with edges weighted by the distance can be very informative for the initial results inspection (as on the screenshot of the graph below), we have decided to perform the exploratory analysis using “more directed” tools.



The figure below depict the distance matrix between the languages:

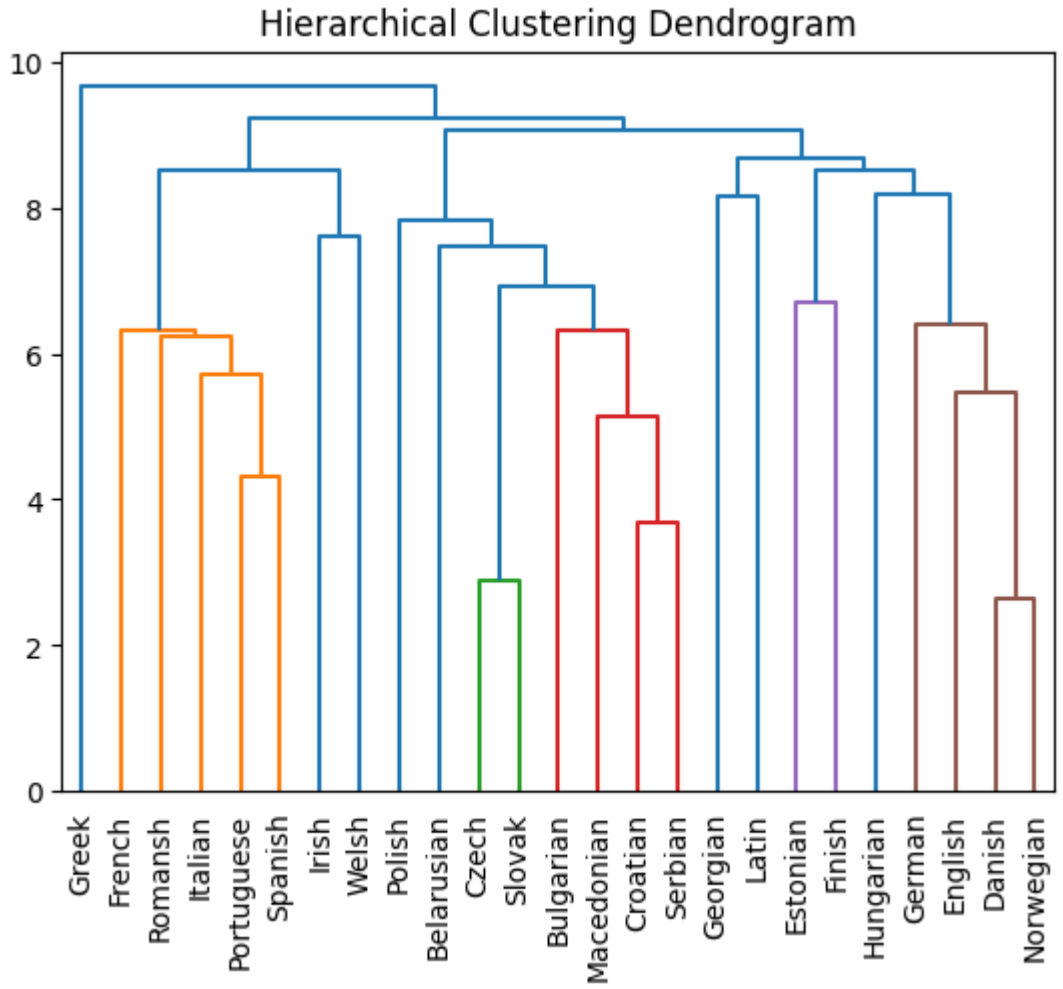


Greek differs the most from other languages, the smallest distance it got is 8.7 which is rather far. Welsh and Georgian, which are also languages from a single language group (in our dataset), get much closer to other languages.

Countries that are close to each other (Norwegian/Danish) and that were historically involved (Croatian/Serbian and Czech/Slovak) get very close, even to 2.6.

Interestingly Polish distance does not show that it's in slavic group, other languages in it are much closer to one another, even those that are close to Poland.

We have additionally made an attempt in clustering the languages according to their pairwise distances with the use of the agglomerative clustering algorithm which recursively merges pairs of clusters of sample data by minimizing the maximum distance between the observations in each cluster. The dendrogram below depicts the cluster formation.



This clustering clearly displays language groups combined with geographical distance of the countries. But as we have seen in Levenstein Polish is not combined with any other language, even Belarusian which is really close not only on the graph but also geographically and language alike.