Text-based Similarity Analysis of Languages

Jakob Mass (1st yr Software Engineering Master's Curriculum), Johann Lutterodt (Exchange student)
Institute of Computer Science, Faculty of Mathematics and Computer Science, University of Tartu

Using lexical distance and the Bible, we explored the similarity of 20 European languages and devised an approach that can successfully recognize similarity between languages with results comparable to established language family groups. Additionally, the analysis tool has been implemented as an easy-to-use web application.

The Data

From a variety of online resources, we devised a dataset consisting of the first chapter from the Book of Genesis from the Old Testament, in 20 European languages.

The Bible was chosen as it is a piece of text which is freely available in a plethora of languages. In addition, the verses of the Bible provide the quality of having subparts of the text which are semantically equivalent.

from
Í upphafi skapaði Guð himin og jörð.
to
i upphafi skapadi gud himin og jord

Preprocessing

In order to make the languages comparable, we got rid of punctuation, newline- and excessive whitespace characters. In addition, we transformed all characters to lowercase since the distance measures are case sensitive.

We replaced all language-specific special characters with similar, more general characters a process known as transliteration (Latin to ASCII characters) . For example, õ,ä,ö,ü are turned into o,a,o,u.

Method

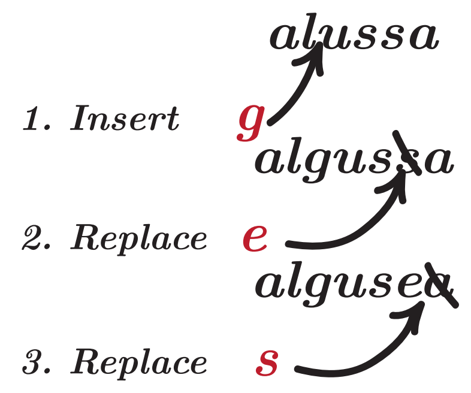
In order to compare any pair of languages, we combined two string distance measuring algorithms: Damerau-Levenshtein distance and the length of the longest common subsequence to create an ensemble where 40% of the final distance score is attributed to the Damerau-Levenshtein measure and 60% is attributed to LCS.

Damerau-Levenshtein distance

The Levenshtein distance, also known as edit distance, is determined by the number of single character edit operations needed to change one string into another. A smaller score indicates larger similarity.

From "alussa" to "alguses" with Levenshtein:

Total distance = 3



The classic Levenshtein distance uses the following operations: insertions, deletions or substitutions. However, we used a modified version of this algorithm, called the Damerau–Levenshtein distance, which also allows the swapping of two neighbouring characters.

To pervent dissimilar, short strings getting good scores through coincidence, we normalized the distance using the length of the longer string of the pair being compared. This prevents distances between small strings being as significant as distances between longer strings.

Short strings have higher chance to be similar through coincidence:

Levenshtein("abc","xyz")= 3

Thus, length-based normalization is needed!

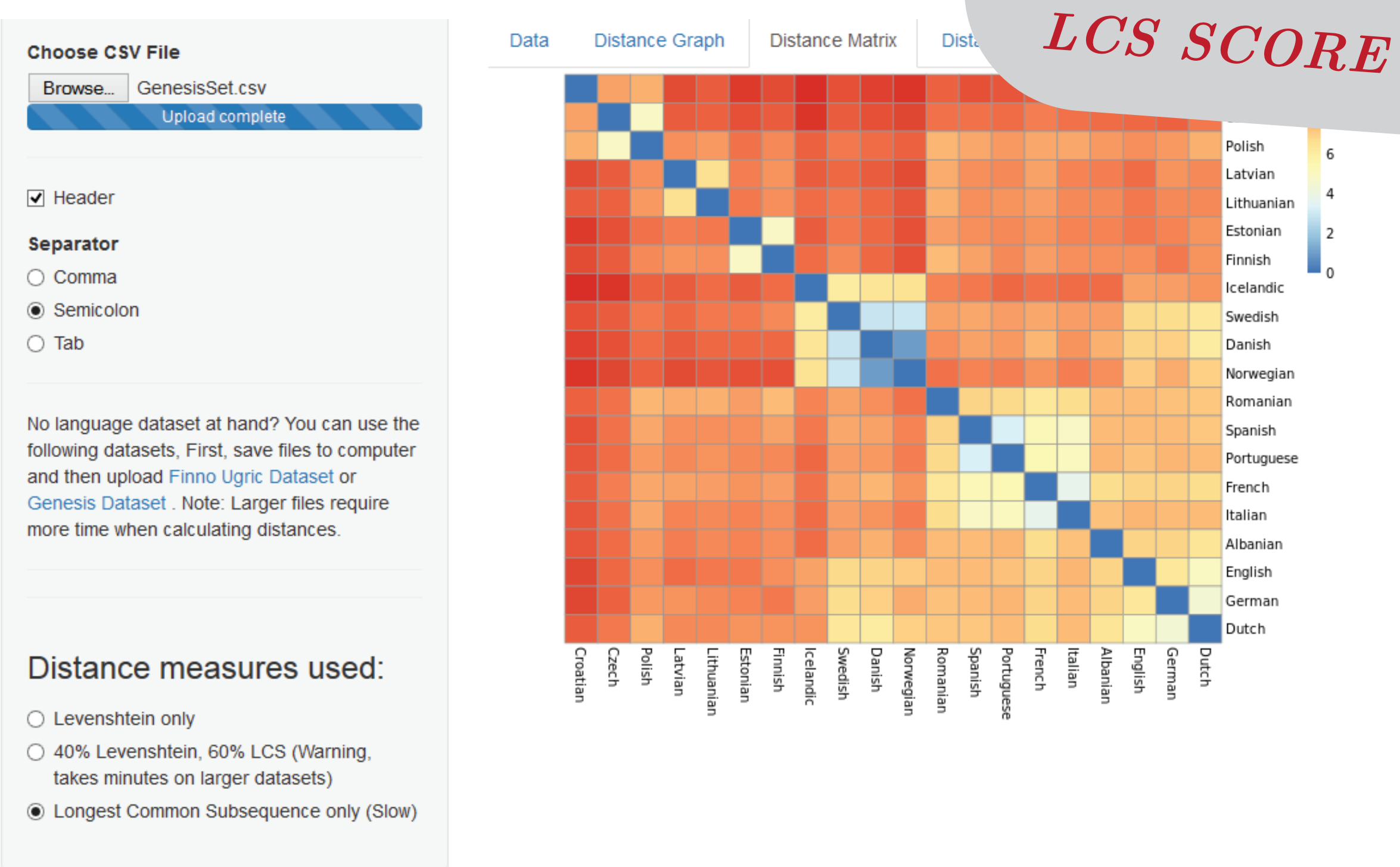
Longest Common Subsequence

This algorithm, given multiple sequences, finds the longest subsequence which is contained in all of the given sequences. The important difference between finding common subsequences and substrings is the fact that subsequences do not need to be consecutive.

In our case, the length of the longest common subsequence (LCS) indicates the similarity of two languages. The bigger the length, the more similar the languages.

x = *in the beginning*
y = *in den biginne*
$LCS(x,y)$ = *inebginn*

LCS SCORE = 8



Choose CSV File
Browse... GenesisSet.csv
Upload complete

☑ Header

Separator
○ Comma
● Semicolon
○ Tab

No language dataset at hand? You can use the following datasets, First, save files to computer and then upload Finno Ugric Dataset or Genesis Dataset . Note: Larger files require more time when calculating distances.

Distance measures used:
○ Levenshtein only
○ 40% Levenshtein, 60% LCS (Warning, takes minutes on larger datasets)
● Longest Common Subsequence only (Slow)

The web-app version of our approach, which provides multiple graphical views on the results of the lexical analysis.

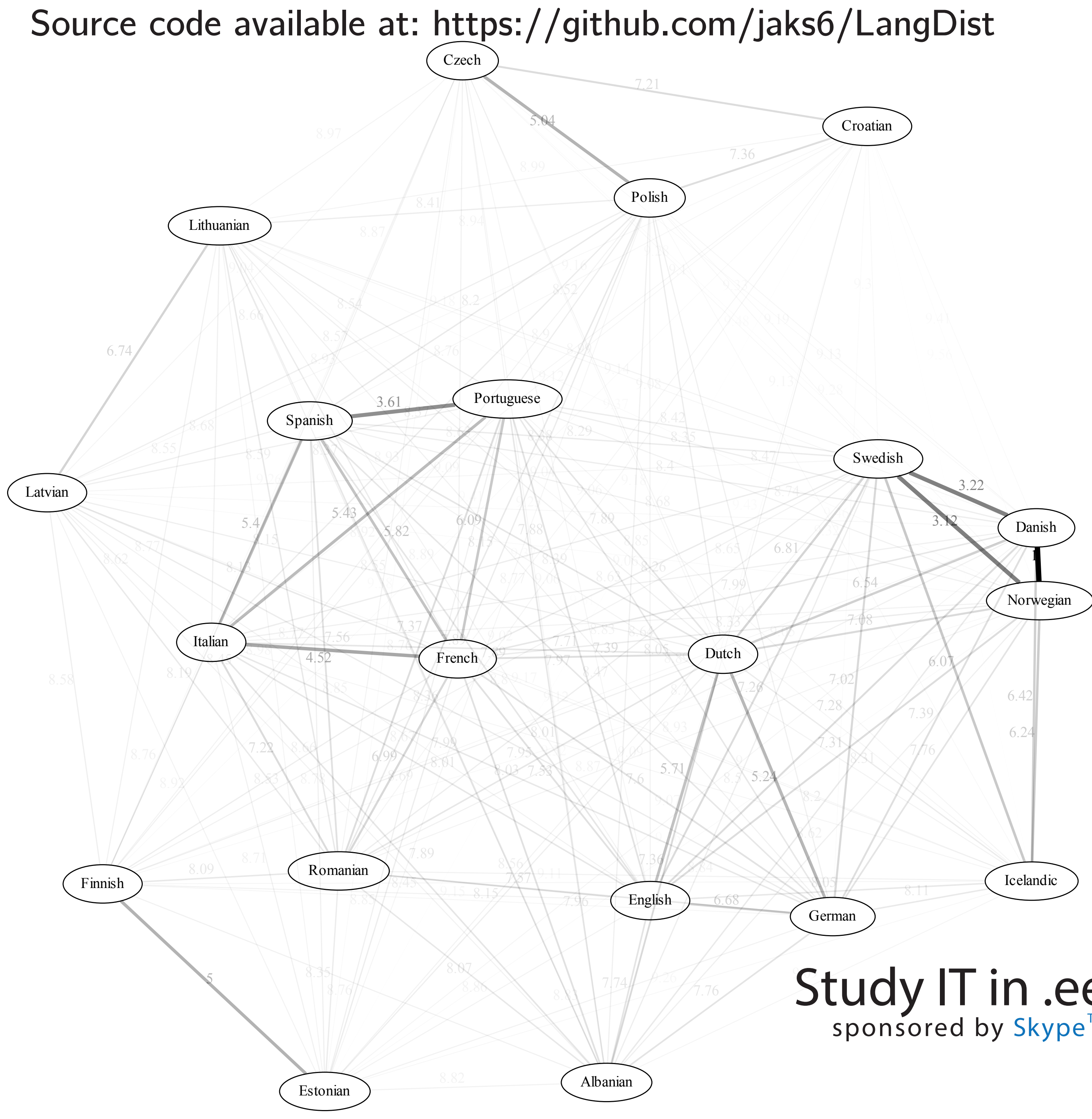
Results

Using our combined distance measure, we were able to meaningfully represent the data on a graph representing the distances and relations between the 20 languages in our dataset. The graph distinguishes 5 main language groups, for example, the Finno-Ugric family or the Romance languages.

After finding the results satisfactory, we organized our approach into an easy-to-use web application which can be found at:
http://kodu.ut.ee/~jaks/langdist

The application is based on R and Shiny and can be used with arbitrary inputs. It performs our analysis on the provided input and provides additional graphical views such as heatmaps and dendograms.

Source code available at: https://github.com/jaks6/LangDist



UNIVERSITY OF TARTU 1632

Study IT in .ee
sponsored by Skype™