Introduction

Vandalism detection in OpenStreetMap

Author Davide Ferrara

Topic: Software

Faculty: Department of Computer Sciences

Address: Eastern Switzerland University of Applied Sciences Campus Rapperswil-Jona

Abstract

This work is divided into two parts: Part 1 deals with the compilation of a data collection of localities (place names) in German-speaking countries that have been identified as "bad edits" in OpenStreetMap [1] (OSM). Part 2 consists of programming a function in Python, that check the similarity of entered words with words from the collect "Bad Words" lists to determine whether the change in OSM is a potentially real locality or not. In the first part, several data collections were compiled in German (de) and English (en). Approx. 270 words with unappropriated expressions were compiled from the "deTenTen" corpus (de) of Sketchengine [2] and approx. 1600 words from Kaggle (en) [3]. 20 bad edits were collected from OSM with the help of the online tool OSMCha [4] and outputs so-called "changesets" [5]. The very low number of 20 bad edits found is mainly due to the fact that OSM has mechanisms and tools that catch the majority of unwanted bad edits [6] before they are documented. For testing and training purposes, an additional 1200 German-language names were extracted from OSM using the OSM online tool Overpass-Turbo [7]. In the second part, various algorithms were compared with string metrics that return normalized values between 0.0 (unequal) and 1.0 (equal). The algorithms Jaro-Winkler (JW) [8], Levenshtein (LV) [9], Smith-Waterman (SM) [10] and Cosine Distance (CD) [11] were implemented. The most suitable algorithm was evaluated and a suitable threshold value was determined heuristically. These are the results for four selected test metrics [12]: F1 score: SM 0.987, CD 0.989, JW 0.99, LV 0.99; "False Positive Rate": SM 0.936, CD 0.848, JW 0.657, LV 0. 0; "False Discovery Rate": SM 0.682, CD 0.888, JW 0.963, LV 0.997; "True Negative Rate": SM 0.84 CD 0.92 JW 0.92 LV 0.84. The "True Negative Rate" and above all the F1 scores are similar for all algorithms. The LV algorithm performs best for the False Discovery Rate and the False Positive Rate. The SM algorithm does not perform as well when recognizing word similarities. This is probably because it was developed to find long DNA sequences. This is probably because it was developed to find long DNA sequences. The CD algorithm also appears to be more suitable to compare whole sentences. The JW algorithm - a further development of the LV algorithm - performs slightly better than the CD and SM algorithms. In contrast to the LV algorithm, the JW algorithm places more emphasis on identical word beginnings, which is suitable for longer word strings or texts. The LV algorithm determines how many mutations must be carried out in order to arrive at the word to be compared. It has advantages for short strings, which seems to be most suitable for the problem at hand. With approx. 1300 test words and a heuristically determined threshold value of 0.86, it delivered all true positives and only 4 false negatives.

Management summary

With OSM an editable map is available which are developed with a big community [13]. Because everyone which is registered is able to edit this map, Bad Edits can be created. An already installed profanity filter from OSM prevents most of these Edits to get to the user or the public. This project should collect dataset for being able to prevent further inappropriate naming of local names. These will be archived by a function with an integrated algorithm comparison to find every word, which matches for being too similar to a word of the database.

The compilation of a data collection of localities (place names) in German-speaking countries that have been identified as "bad edits" in OSM are quite time intensive. For this task the object is to have a workable database.

Part 2 consists of programming a function in Python, that check the similarity of entered words with words from the collect "Bad Words" lists to determine whether the change in OSM is a potentially real locality or not.

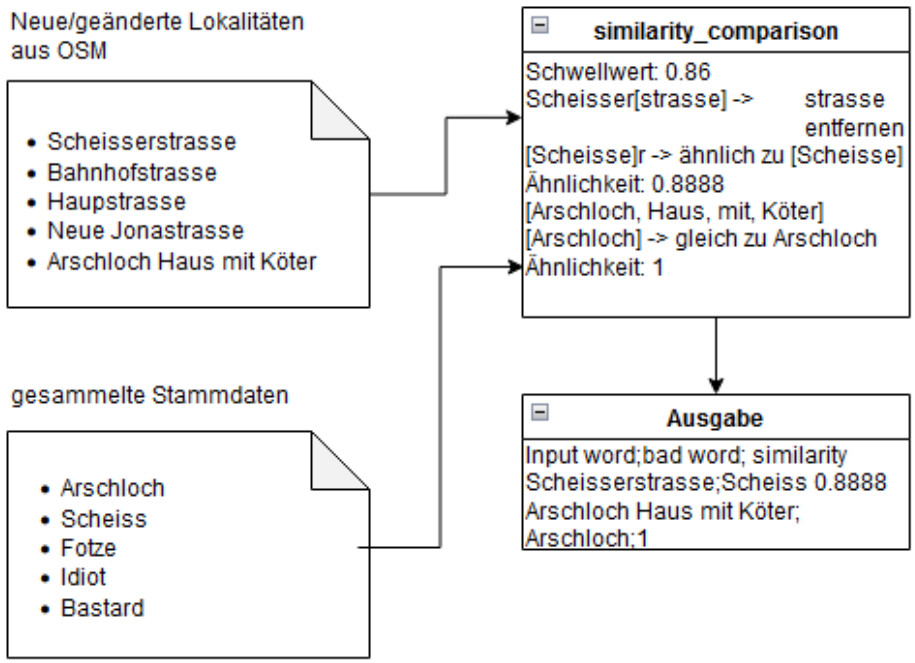

Figure 1: data flow diagram: Input data are testing words (up left) and the dataset collection with bad words (down left) which are going through the function and give the output

Several data collections were compiled in German (de) and English (en). Approx. 270 words with unappropriated expressions were compiled from the "deTenTen" corpus (de) of Sketchengine and approx. 1600 words from Kaggle (en). 20 bad edits were collected from OSM with the help of the online tool OSMChas and outputs so-called "changesets". The very low number of 20 bad edits found is mainly due to the fact that OSM has mechanisms and tools that catch the majority of unwanted bad edits before they are documented. For testing and training purposes, an additional 1200 German-language names were extracted from OSM using the OSM online tool Overpass-Turbo. For the second task, various algorithms were compared with string metrics that return normalized values between 0.0 (unequal) and 1.0 (equal). The algorithms Jaro-Winkler (JW), Levenshtein (LV), Smith-Waterman (SM) and Cosine Distance (CD) were implemented. The most suitable algorithm was evaluated and a suitable threshold value was determined heuristically.

```
/*
Hilfe siehe https://dev.overpass-api.de/overpass-doc/de/
und https://giswiki.hsr.ch/Overpass_API
*/
[out:csv(::type,::id,name,"name:en")];
//[out:json];
(
  nwr["name"]["name:en"](46.6645,7.3114,47.8814,9.9042);
  //nwr["name"]["name:en"]({{bbox}});  // interaktiv in Overpass Turbo
);
//out body; >; out skel qt;
out center;
```

```
node    197266469   Steffisburg Steffisburg
node    240025182   Zürich  Zurich
node    240035066   Graubünden/Grigioni/Grischun    Grisons
node    240079173   Thurgau Thurgau
node    241748857   Burg Rötteln    Rötteln Castle
node    280528109   Antikensammlung University of Berne – collection of
node    286879769   Rathaus Bar Rathaus Odeon
node    297764876   Stollenrain "Arlesheim, Stollenrain"
node    303094717   BW-Bank BW-Bank
node    311172753   Holzingers Pavillion    Aach Pavillon
node    313870420   Zürich  Zurich
node    313871105   Basel-Stadt Basel-City
node    385397241   Platz des Feuers    Place of fire Sihlfeld cemetery
node    390456858   Platz der Erinnerung    Place of Remembrance Sihlfe
node    411301584   Parkhaus Zentrum    Parking city centre
node    411872585   Garage Gut Bad Ragaz    BP
node    413741627   Museggmauer Musegg Wall
node    415411479   Weggis  Weggis
node    430269020   Platz der Skulpturen    Sculpture Square Sihlfeld C
node    430281547   Urnenhain Platz Friedhof Sihlfeld   Urn Grove Place
node    430282196   Platz der Gemeinschaft Friedhof Sihlfeld    Communi
node    451722224   Museum Rietberg Rietberg Museum
node    460886548   Avia Tankstelle Avia gas station
node    485569961   Botschaft Argentinien   Embassy of Argentina
node    485569962   Botschaft Malaysia  Embassy of Malaysia
node    534632472   Botschaft Portugal  Embassy of Portugal
node    534632474   Botschaft Mexiko    Embassy of Mexico
```

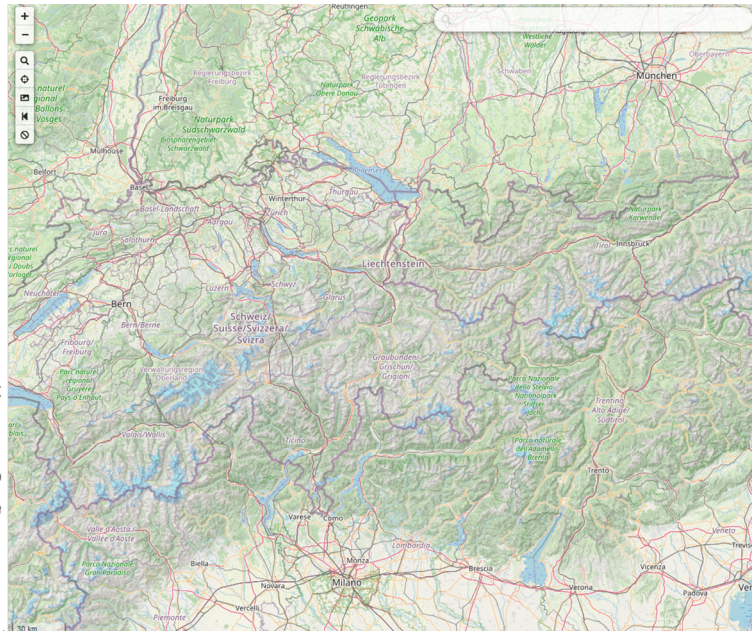

Figure 2: OSM Overpass turbo tool: you see the used query for the names of the mostly german speaking region

	Jaro Winkler		Levenstein		Smith Waterman		Cosinus Distanz	
	Good Words	Bad Words	Good Words	Bad Words	Good Words	Bad Words	Good Words	Bad Words
Median	0.87777778	1	0	1	1	1	0.888888889	1
Mittelwert	0.88263397	0.98704174	0	0.98009861	0.97665048	0.99266373	0.88883195	0.98795768
Differenz: Median - Mittelwe	-0.00485619	0.01295826	0	0.01990139	0.02334952	0.00733627	5.69385E-05	0.01204232
False Positiv:	44		0		309		128	
True Positiv:	1250		1294		985		1166	
False Negativ	2		4		4		2	
True Negativ	23		21		21		23	
Precision	0.99840256		0.99691834		0.99595551		0.998287671	
Recall	0.98193244		0.98403042		0.97912525		0.980656013	
F1 score	0.99009901		0.99043245		0.98746867		0.989393297	
False Positiv Rate	0.65671642		0		0.93636364		0.847682119	
False Discovery Rate	0.9632		0.99690881		0.6822335		0.888507719	
True Positve Rate	0.92		0.84		0.84		0.92	

Figure 3: for measuring the dataset collection version is from 07.12.2023. It used good words for checking if the true/false positives and bad words (fictive testing words) for true/false negatives These are the results for four selected test metrics for the diagram from the data from the analysis above:

Metric	SM	CD	JW	LV
F1 Score	0.98	0.98	0.99	0.99
False Positive Rate	0.94	0.85	0.66	0.0
False Discovery Rate	0.68	0.89	0.96	0.99
True Negative Rate	0.84	0.92	0.92	0.84

Table 1: Metrics

The "True Negative Rate" and above all the F1 scores are similar for all algorithms. The LV algorithm performs best for the False Discovery Rate and the False Positive Rate.

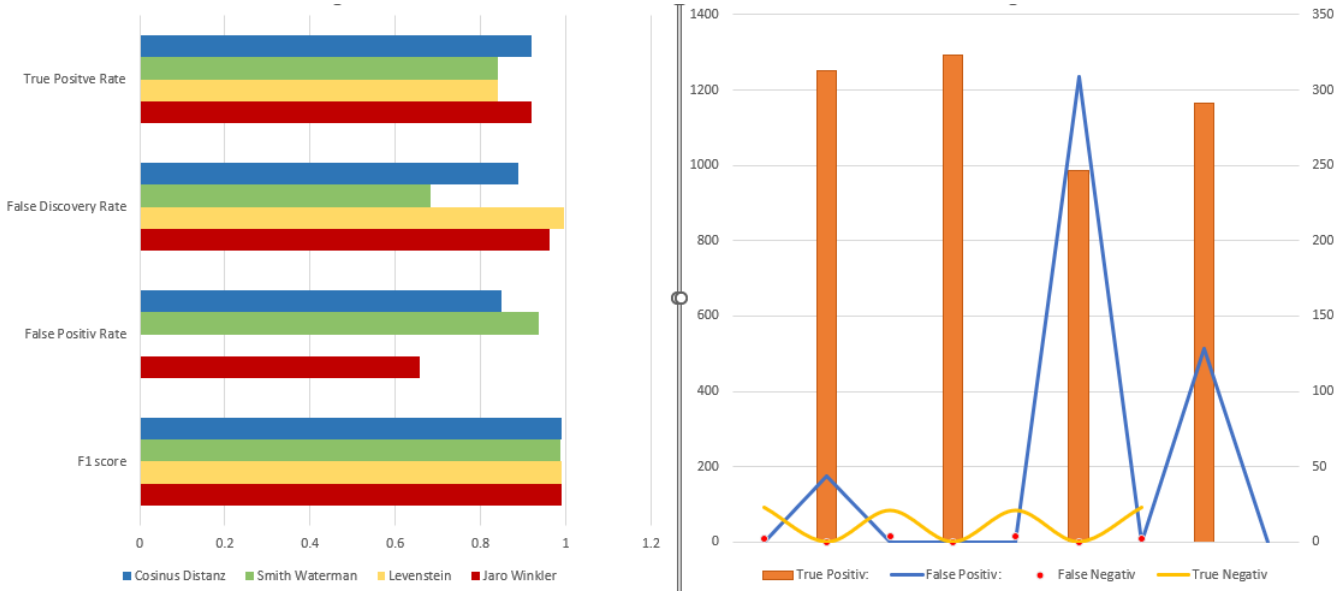

Figure 4: graphics of the metrics: left are the algorithm paired to 4 of the more important metrics. On the right all the parameter, except the true positive, use the right secondary axes for scale.

The SM algorithm does not perform as well when recognizing word similarities. This is probably because it was developed to find long DNA sequences. The CD algorithm also appears to be more suitable for comparing whole sentences. The JW algorithm—a further development of the LV algorithm—performs slightly better than the CD and SM algorithms. In contrast to the LV algorithm, the JW algorithm places more emphasis on identical word beginnings, which is suitable for longer word strings or texts. The LV algorithm determines how many mutations must be carried out to arrive at the word to be compared. It has advantages for short strings, which seem to be most suitable for the problem at hand. With approximately 1300 test words and a heuristically determined threshold value of 0.86, it delivered all true positives and only 4 false negatives.

The main reason for this project is to have a function that identifies bad edits more accurately. With this objective in mind, it is quicker for the user to manually go through a shorter list of only names and evaluate if this was a legitimate edit or not. A larger dataset for bad edits would be the first step for improvement. If more data can be compared, the program can achieve better results. For further development, a more precise analysis must take place to have a better-responding algorithm to unknown data. Comprehensibly, the program does not have good performance. Optimizing the function would be crucial for evaluating large datasets. Further, one of the optional goals was to implement the function in a Flask service. With that in mind, additional resources could be given for quicker calculations.

Content

Chapter 1. Introduction

OpenStreetMap is an opensource available Map which can be used for different purposes. For example the swiss emergency services [14] uses this map and with that everyone will secure a certain integrity of the map. To prevent having to rewrite names, which were written for the sole purpose of vandalizing an openly available map or in the best case to correct accidentally wrongly written names an idea to prevent it was thought of. The idea is to create a more precise sorting system for bad language which will check a list of newly edits and inform editors of the map easier and quicker, which changes should be privatised to manually correct for possible Bad Edits.

1.1. Assignment

The first Goal is to create a data collection, which includes different categories: real data from past vandalised act from OSM, real data from OSM containing the momentary used names on the map, generated testing bad language words and a collection of real data from bad language. The second Goal is to create a vandalism detection tool, which can accurately evaluate, which words should be in use and which shouldn’t. The challenge here is to succeed tools, which are already in daily use by the OSM profanity prevention systems. Optionally, the service could be distributed by flask.

1.2. Basic Condition

This thesis was done as a semester assignment (Studienarbeit) for a workload of 240h for a reward of 240 ETCS credits.

1.3. State of the art

A paper called "Towards Automatic Vandalism Detection in OpenStreetMap" [15] from Geoinformatics Research Group, Department of Geography, University of Heidelberg, has released a paper that shows, how vandalism in general can be mitigated in OSM. The sort of vandalism that this project try to prevent is profanity for naming edits from user. In their paper they describe "inappropriate use of automated edits (bots) in the data" which would be quickly detected by my tool for specifically inappropriate changes in the names. But that isn’t exactly my Goal. Mine is towards detecting name edits, which were vandalised. So getting my detection tool will concentrate only on what are nodes in the OSM.

Chapter 2. OpenStreetMap

OpenStreetMap is a project that creates an openly editable world map. A community is filling the map with all the local information as a collective effort. Everyone can join to help develop the map. In Switzerland, there are about 2,000,000 nodes [16] and each one of them has a local name disposed on the map. There is also a profanity filter in place to prevent vandalism, such as provocative naming of objects and drawing of inappropriate images for a specific location.

2.1. Tools

There were used some tools to making the process of collection data easier.

2.1.1. OSM Overpass Turbo

OSM Overpass provides the possibility of data extrapolation with different possible query languages. For this tool, we are using its Rest API to send a request to get the local name information. The query we are using will be the same, but the user could edit their coordinates to get the desired result.

2.1.2. OSMCha

OSMCha is a tool to search changesets. Because there are sets which are tagged with profanity of different kinds, it reduces the work of finding the right changesets to look at. It accelerates the work of finding the right data.

Chapter 3. Research and data collection

For the first goal a dataset had been collected. After researching in the world wide web not a lot of significant cases were found.

3.1. Bad Language

To define bad language in the context of this tool, a paraphrase similar to this one could describe it: "An edit of a location name that has been modified to match a ridiculed name." There was a case of New York being named "Jewtropolis" [17], which is obviously meant to be insulting but wouldn’t be automatically detected as one due to the nature of names that use "jew" and "tropolis" independently (for example: Jewtropolis used in names and are independently not problematic).

3.2. Data collection

The first goal was to create a data collection that can be used to test the algorithm on it. Additionally, it would prove that vandalism acts are common and needed in OSM to be addressed. Filtering profanity was already discussed in this OSM conference [18]:". They implemented an LLM to create a filtering system for the whole map. It’s precise and detects about 92% of all errors, including wrong mapping. My project is specific for the evaluation of words. Therefore, its target was to be a little better than that.

For every dataset, there are two: one in German and one in English.

3.2.1. Data from OSM

These are the datasets which were provided or found. It is the foundation of this project, to have a base of bad language word to be able to compare them to a given list of words to be checked.

3.2.2. Bad Edits example from OSM

These are the Bad Edits found from the real data on OSM.

1	Table/File Name	OSM_example_en
2	Format	XLSX
3	License / Usage Rights	ODbL
4	Coordinate Reference System	N/A
5	Quality	
5.1	Accuracy/Resolution	N/A
5.2	Timeliness/Update Status	12/14/2023
5.3	Completeness	No
5.4	Logical Consistency	First column words, Second column link to OSM
5.5	Transparency/Origin/Processing/Contact	Extracted from OSM:
5.6	Data Schema	text/varchar(1024)

Table 2: OSM_example_en, meta-information

1	Table/File Name	OSM_example_de
2	Format	XLSX
3	License / Usage Rights	ODbL
4	Coordinate Reference System	N/A
5	Quality	
5.1	Accuracy/Resolution	N/A
5.2	Timeliness/Update Status	12/14/2023
5.3	Completeness	No
5.4	Logical Consistency	First column words, Second column link to OSM
5.5	Transparency/Origin/Processing/Contact	Extracted from OSM:
5.6	Data Schema	text/varchar(1024)

Table 3: OSM_example_de, meta-information

3.2.3. Real Testing words

These are the datasets which were generated from the real existent data from the OSM. The Overpass tool was excellent for this task.

1	Table/File Name	overpass_en
2	Format	XLSX
3	License / Usage Rights	ODbL
4	Coordinate Reference System	N/A
5	Quality	
5.1	Accuracy/Resolution	N/A
5.2	Timeliness/Update Status	12/14/2023
5.3	Completeness	100%
5.4	Logical Consistency	Yes, all words
5.5	Transparency/Origin/Processing/Contact	Extracted from OSM Overpass Turbo: Query link
5.6	Data Schema	text/varchar(1024)

Table 4: overpass_en, meta-information

1	Table/File Name	overpass_de
2	Format	XLSX
3	License / Usage Rights	ODbL
4	Coordinate Reference System	N/A
5	Quality	
5.1	Accuracy/Resolution	N/A
5.2	Timeliness/Update Status	12/14/2023
5.3	Completeness	No
5.4	Logical Consistency	First column words, Second column link to OSM
5.5	Transparency/Origin/Processing/Contact	Extracted from OSM Overpass Turbo: Query link
5.6	Data Schema	text/varchar(1024)

Table 5: overpass_de, meta-information

3.2.4. Existing lists of bad language for comparison

These are found list with bad words which are central. These were used for the algorithm to have a parameter to compare.

1	Table/File Name	bad-words
2	Format	CSV
3	License / Usage Rights	CC0: Public Domain
4	Coordinate Reference System	N/A
5	Quality	
5.1	Accuracy/Resolution	N/A
5.2	Timeliness/Update Status	2017
5.3	Completeness	N/A
5.4	Logical Consistency	Yes, all words
5.5	Transparency/Origin/Processing/Contact	33%
5.6	Data Schema	text/varchar(1024)

Table 6: bad-words, meta-information

1	Table/File Name	thesaurus_preloaded_detenten20_rft3_20230608162039
2	Format	XLSX
3	License / Usage Rights	Sketch engine
4	Coordinate Reference System	N/A
5	Quality	
5.1	Accuracy/Resolution	N/A
5.2	Timeliness/Update Status	08/06/2023
5.3	Completeness	N/A
5.4	Logical Consistency	Yes, all words
5.5	Transparency/Origin/Processing/Contact	
5.6	Data Schema	text/varchar(1024)

Table 7: deTenTen 20, meta-information

3.2.5. Testing words

These are the fiction words used to test the algorithm. They were generated mostly by ChatGPT and the rest through creative thinking.

1	Table/File Name	testing_words_en
2	Format	XLSX
3	License / Usage Rights	N/A
4	Coordinate Reference System	N/A
5	Quality	
5.1	Accuracy/Resolution	N/A
5.2	Timeliness/Update Status	12/14/2023
5.3	Completeness	No
5.4	Logical Consistency	Yes, all words
5.5	Transparency/Origin/Processing/Contact	Generated from Chat GPT, prepared in xlsx
5.6	Data Schema	text/varchar(1024)

Table 8: testing_words_en, meta-information

1	Table/File Name	testing_words_de
2	Format	XLSX
3	License / Usage Rights	N/A
4	Coordinate Reference System	N/A
5	Quality	
5.1	Accuracy/Resolution	N/A
5.2	Timeliness/Update Status	12/14/2023
5.3	Completeness	No
5.4	Logical Consistency	Yes, all words
5.5	Transparency/Origin/Processing/Contact	Generated from Chat GPT, prepared in xlsx
5.6	Data Schema	text/varchar(1024)

Table 9: testing_words_de, meta-information

Chapter 4. Different language attributes

Some language have specific attributes that can be addressed to get a better result, for sorting or evaluating their meaning. In order to produce better quality analysis, there was a bit of research involve to get to most important features.

4.1. German

The German language has some specific language features that are quite unique. One of them is the use of compound words, which means that two words get concatenated together to one (Main Street would be "Hauptstrasse" and not "Haupt Strasse").

An explanation form the wikipedia page[19]: "In linguistics, a compound is a lexeme (less precisely, a word or sign) that consists of more than one stem. Compounding, composition, or nominal composition is the process of word formation that creates compound lexemes. Compounding occurs when two or more words or signs are joined to make a longer word or sign. A compound that uses a space rather than a hyphen or concatenation is called an open compound or a spaced compound; the alternative is a closed compound."

A compound string splitter [20] was experimented with. It wasn’t accurate enough for the purpose, and due to a lack of resources, it was abandoned.

4.2. English

One of the interesting things when working with strings is that a street name like "Main Street" is one string. With that, we get multiple words in one string.

Fortunately, an existing function `split()` exists, which separates a string by white spaces into a list of substrings. With this method, the evaluation can take part for every word alone and get more precise results.

Chapter 5. Algorithms

All these algorithms have the function to evaluate how similar a string is to another string. This involves spell checking, length checking, and some kind of metric to give weight to important parameters. Furthermore, for this particular function, it needed to be normalized to a float number between zero and one to be compared to the other algorithms.

5.1. Jaro Winkler

This algorithm is from 1989 and uses different attributes to get to its final result. It compares the strings, their length to each other, and how many transpositions are made. In addition, it has an added prefix scaling factor, which gives favorable calculations for similar sequences at the beginning of the strings.

Here is the definition from wikipedia [19]: "In computer science and statistics, the Jaro–Winkler similarity is a string metric measuring an edit distance between two sequences. […] The Jaro–Winkler distance uses a prefix scale p p which gives more favourable ratings to strings that match from the beginning for a set prefix length l.

The higher the Jaro–Winkler distance for two strings is, the less similar the strings are. The score is normalized such that 0 means an exact match, and 1 means there is no similarity. The original paper actually defined the metric in terms of similarity, so the distance is defined as the inversion of that value (distance = 1 - similarity).

The implementation used is from the Institute for Software from my university OST at the Rapperswil-Jona campus.

5.2. Levenshtein

Levenshtein is an algorithm, which describe how many changes a word or a string have to be made compared to another to get the same word. For example: Hallo and Hello needs just one transposition to be the exact same word.

Hers is the definition from wikipedia [19]: "Informally, the Levenshtein distance between two words is the minimum number of single-character edits (insertions, deletions or substitutions) required to change one word into the other. It is named after the Soviet mathematician Vladimir Levenshtein, who considered this distance in 1965.

Levenshtein distance may also be referred to as edit distance, although that term may also denote a larger family of distance metrics known collectively as edit distance. It is closely related to pairwise string alignments."

The code provided for this is from this webpage: https://copyprogramming.com/howto/damerau-levenshtein-edit-distance-in-python with some changes to be better readable and adapt to the value on the end between zero and one. More details will be given in chapter 7 Implementation and Code Base.

5.3. Smith Waterman

The smith waterman algorithm was designed in 1970 to helping to analyze gene sequences.

Here is a definition form wikipedia [19]: "The Smith–Waterman algorithm performs local sequence alignment; that is, for determining similar regions between two strings of nucleic acid sequences or protein sequences. Instead of looking at the entire sequence, the Smith–Waterman algorithm compares segments of all possible lengths and optimizes the similarity measure.

The algorithm was first proposed by Temple F. Smith and Michael S. Waterman in 1981. Like the Needleman–Wunsch algorithm, of which it is a variation, Smith–Waterman is a dynamic programming algorithm. As such, it has the desirable property that it is guaranteed to find the optimal local alignment with respect to the scoring system being used (which includes the substitution matrix and the gap-scoring scheme). The main difference to the Needleman–Wunsch algorithm is that negative scoring matrix cells are set to zero, which renders the (thus positively scoring) local alignments visible. Traceback procedure starts at the highest scoring matrix cell and proceeds until a cell with score zero is encountered, yielding the highest scoring local alignment. […]

For our purpose, it was crucial to find one code fragment where the function's return could be reduced to a float result between 0 and 1. For this particular algorithm, it's possible to get a result over 1 because it checks sequences of words, and these can be found multiple times in a string comparison.

5.4. Cosine Distance

This algorithm comes from the data analysis and is used to compare non-zero two vectors to declare their similarity to each other. Normally it is used to get a comparison of whole texts.

Here is a description from the wikipedia [19]: "In data analysis, cosine similarity is a measure of similarity between two non-zero vectors defined in an inner product space. Cosine similarity is the cosine of the angle between the vectors; that is, it is the dot product of the vectors divided by the product of their lengths. It follows that the cosine similarity does not depend on the magnitudes of the vectors, but only on their angle. The cosine similarity always belongs to the interval [¬ 1 , 1] . [-1,1].} For example, two proportional vectors have a cosine similarity of 1, two orthogonal vectors have a similarity of 0, and two opposite vectors have a similarity of -1. In some contexts, the component values of the vectors cannot be negative, in which case the cosine similarity is bounded in [0 , 1] [0,1].

For example, in information retrieval and text mining, each word is assigned a different coordinate and a document is represented by the vector of the numbers of occurrences of each word in the document. Cosine similarity then gives a useful measure of how similar two documents are likely to be, in terms of their subject, and independently of the length of the documents."

Unfortunately, this algorithm is designed to compare more than just a few letters. To adapt to these circumstances, in the implementation, there is a short helper function to create vectors out of single letters and not words. Furthermore, the vectors must be normalized to get a useful score.

Chapter 6. Algorithm Analysis

After having a good comparison of four widespread algorithms, they must be analyzed to define their usefulness. One aspect for a better analysis is already in the program: They are all normed to get a value between 0 and 1. With that in mind, an easy and compact code was written to compare their float numbers as similarities.

6.1. Metrics

The f1-score metric is central with the implication of some individual part of it. Additionally, some other common used metric where compared to have a more distinguished view. Following metrics are used:

Metric	Description
True Positive	Passing words rightfully examined as acceptable words.
False Positive	Passing words wrongfully examined as unacceptable words.
True Negative	Failing words rightfully examined as unacceptable.
False Negative	Failing words wrongfully examined as acceptable.
Precision	Ratio of true positive to the sum of true positive and false positive.
Recall	Ratio of true positive to the sum of true positive and false negative.
False Positive Rate	Ratio of false positive to the sum of false positive and true positive.
True Negative Rate	Ratio of true negative to the sum of true negative and false positive.
False Discovery Rate	Ratio of true positives to the sum of true positives, false positives, and false negatives.
F1 Score	Ratio between precision and recall, indicating the precision of a function.

Table 10: Explanation Metrics

The true/false positives/negatives are disposed in the analysis to set a better understanding of the comparison of ratios to the belonging size of the dataset. The second group of metrics is chosen to give a more differentiated view of the important f1-score metric. The f1-score metric is almost indistinguishable in the selection of the algorithm used in this project. Therefore, a better approach in analyzing must be provided through more and different metrics.

6.2. Analysis

The four different algorithms were run through the program. The dataset used was the one from 07.12.2023. The output data generated were put for easier access and editable in an Excel sheet. From there, the median and the average were calculated between one set of only testing words which should fail the test and one dataset of only passing testing words. A number between the two sets was chosen for further recursion. Due to a lack of time, another analysis with the new data couldn't be done.

	Jaro Winkler		Levenstein		Smith Waterman		Cosinus Distanz	
	Good Words	Bad Words	Good Words	Bad Words	Good Words	Bad Words	Good Words	Bad Words
Median	0.87777778	1	0	1	1	1	0.888888889	1
Mittelwert	0.88263397	0.98704174	0	0.98009861	0.97665048	0.99266373	0.88883195	0.98795768
Differenz: Median - Mittelwe	-0.00485619	0.01295826	0	0.01990139	0.02334952	0.00733627	5.69385E-05	0.01204232
False Positiv:	44		0		309		128	
True Positiv:	1250		1294		985		1166	
False Negativ	2		4		4		2	
True Negativ	23		21		21		23	
Precision	0.99840256		0.99691834		0.99595551		0.998287671	
Recall	0.98193244		0.98403042		0.97912525		0.980656013	
F1 score	0.99009901		0.99043245		0.98746867		0.989393297	
False Positiv Rate	0.65671642		0		0.93636364		0.847682119	
False Discovery Rate	0.9632		0.99690881		0.6822335		0.888507719	
True Positve Rate	0.92		0.84		0.84		0.92	

Figure 5: results of the analysis: for measuring the dataset collection version is from 07.12.2023. It used good words for checking if the true/false positives and bad words (fictive testing words) for true/false negatives

6.3. Results

The result of the analysis is quite simple, giving two averages, one for a good and one for a better-performing algorithm. Cosine distance and Smith-Waterman are limited in their usage for this task. Looking at their designed purpose, it is not surprising that there are other better ones. The Jaro-Winkler algorithm distinguishes better between the true positives and false positives but was created to look at longer strings. Its approach to weighing in the prefixes of a word is one of the bigger differences between Jaro-Winkler and the Levenshtein algorithm. But this program placed importance on single words because it results in a more precise evaluation of the strings. Certainly, the Levenshtein is the best-performing algorithm for this intent and the restricted time tied to the project. It is more precise for its purpose than the implemented profanity filter from OSM.

Chapter 7. Implementation and Code base

For this project a whole application wasn't necessary to create, so it was limited to one main function "similarity_comparison".

7.1. Inputs

Same with that, the idea was to give the user the possibility to take a link from Overpass turbo query (de) [21] (en) [22] and put the link into the tool. The request is predefined and needs to be adjusted for the wished coordinates. With that, a connection to the OSM server with the query request would be made, and the rest would be handled by the function. Nevertheless, there is no problem to simply give a list as an input and let the function handle the rest. For English words in the German-speaking parts, this query was used:

```
[out:csv("name:en")];
(
nwr["name"]["name:en"](46.6645,7.3114,47.8814,9.9042);
);
out center;
```

Figure 6: query for Overpass turbo for english names

For local german names in the mostly german speaking parts this query were used:

```
[out:csv(name)];
(
nwr["name"](46.6645,7.3114,47.8814,9.9042);
);
out center;
```

Figure 7: query for Overpass turbo for german words Their output is simple a list of names with name or name:en as title. But the easier way to use the similarity comparison function is to take the link from the query which one could use and then past the link into the overpass function

7.2. Function

The similarity_comparison function is our main function. When it is called, it calls other helper functions to get through the whole process of creating a CSV file for the user. It has 5 parameters: language (German or English), algorithms, the name of the CSV file, the input data, and the threshold value defining when the similarity is enough to label it as problematic. That's how it looks like:

```
def similarity_comparison(data_set, algorithm, excel_name, testing_words, threshold):
    dictionaries_list = create_similarity_dictionaries(data_set, testing_words, algorithm)
    column1 = create_data_column()
    for dictionary in dictionaries_list:
        max_score = calculate_max_score(dictionary)
        filter_max_score(dictionary, max_score, column1, threshold)
    create_csv(excel_name, column1)
```

Figure 8: shows similarity_comparison function Given a list of strings as an input the function will use a chosen algorithm. The algorithm has a predefined list of words comparing with the input data. Given that the Input word could be a multiple word string it gets divided into single words with the .split() function and then checked individually. The procedure will return a list of individual dictionary for every word which was

in the input. The dictionary has, as key, the word out of the unappropriated language list. Values are a tuple with the input word and the value of the similarity, given by the algorithm between the individual input word and the key word.

```
def create_similarity_dictionaries(data_set, input_words, algorithm):
    keys_dictionary = chose_extension_list(data_set)
    input_list = chose_extension_list(input_words)
    similarity_dictionaries = [{}]
    for words in input_list:
        length_name = len(words)
        for sub_word in rd.common_used_suffixes_de:
            mod_word = cut_common_suffixes_de(words, sub_word)
            if length_name != len(mod_word):
                break
        word = mod_word.split()
        for name in word:
            similarity_dictionary = {}
            for dict_key in keys_dictionary:
                similarity_dictionary[dict_key] = (algorithm(dict_key, name), words)
            similarity_dictionaries.append(similarity_dictionary)
    return similarity_dictionaries
```

Figure 9: shows create_similarity_dictionaries function

The easiest way to get the highest score of similarity of each dictionary is to create a panda Series and then get the maximum value out of the of it.

```
def calculate_max_score(input_dictionary):
    data_statistic = []
    for number in input_dictionary.values():
        data_statistic.append(number[0])
    median_values = pd.Series(data_statistic)
    max_score = median_values.max()
    return max_score
```

Figure 10: shows calculate_max_score function

With the maximum score the individual word gets checked to be inserted, if it has this similarity, if it is over the number on which it is identified as bad language and if it already declared in the output dictionary.

```
def filter_max_score(input_dictionary, max_score, raw_data, threshold):
    for index, (key, values) in enumerate(input_dictionary.items()):
        value = values[0]
        input_word = values[1]
        if max_score in values and value > threshold and input_word not in raw_data['Input_Word']:
            raw_data['Input_Word'].append(input_word)
            raw_data['Bad_Word'].append(key)
            raw_data['Similarity'].append(value)
```

Figure 11: shows filter_max_score function

After finishing the last dictionary the output csv is created and wrote with all which is in the output dictionary. The file is saved in the same directory as the working directory with the given name.

For having a better result for the german words, a helper function was implemented to cut the most used post suffixes which would make the comparison of the algorithm more precise. These are the used suffixes:

```
common_used_suffixes_de = ["strasse", "strassen", "gasse", "gassen", "gässli", "platz", "plätzli", "weg", "wege",
                           "heim", "allee"]
```

Figure 12: shows the list of prefixes which gets deleted, when found in input words

7.3. Output

The output is a csv with ";" as a separator and the data is given in the following order per row: The initial input word, the bad word from the collected data and the similarity score.

7.4. Algorithms and Levenshtein

Every algorithm used for the analysis of the datasets was standardized in the sense that each algorithm was forced to accept one input word, which was checked with a bad word from the collected dataset. Additionally, every output of the algorithms produced a float number between zero and one to have an easier approach to compare the data.

The thought of this adapting of algorithms if necessary was to ensure for future development. If someone finds a better or needs to test other algorithms, no changes are needed in the main file.

The only algorithm recommended for use in production is the Levenshtein algorithm, which will be discussed in detail in the following lines.

```
def levenshtein_similarity(input_word, bad_word):
    len_input = len(input_word)
    len_bad_word = len(bad_word)
    # Creates 2D matrix
    matrix = [list(range(len_input + 1))] * (len_bad_word + 1)
    # insert values of the first word
    for word in list(range(len_bad_word + 1)):
        matrix[word] = list(range(word, word + len_input + 1))
    # insert values of the second word
    for word in list(range(0, len_bad_word)):
        for comparison in list(range(0, len_input)):
            if input_word[comparison] == bad_word[word]:
                matrix[word+1][comparison+1] = min(matrix[word+1][comparison] + 1, matrix[word][comparison+1] + 1,
                                                   matrix[word][comparison])
            else:
                matrix[word+1][comparison+1] = min(matrix[word+1][comparison] + 1, matrix[word][comparison+1] + 1,
                                                   matrix[word][comparison] + 1)
    distance = float(matrix[len_bad_word][len_input])
    similarity = 1.0 - distance / max(len_input, len_bad_word)
    return similarity
```

Figure 13: shows the implemented Levenshtein algorithm

The function is called with two parameters: a word to check, which is the input word, and a word to be checked against, which in our case is used from the dataset. The lengths of the inputs are measured because, in the second step of the function, a 2D matrix is created with the lengths of the input in the first row for the next row with the length of the bad word. Then a value insertion of the letters from the input and bad words proceeds. It will generate the Levenshtein distance in the bottom right corner of the matrix. This value needs to be converted to a value from zero to one, which happens in the line with the similarity, where the whole distance is divided by the longer length of the two parameters. The return value is the similarity.

Chapter 8. Conclusion

The project could not be completed entirely. There was significant time pressure, mainly due to the initially unsuccessful search for words for the datasets, which weighed heavily. Additionally, this is the first larger project in the software field that I have tackled alone.

The core goals were achieved. These included providing datasets and implementing a function that works well for comparing words. However, due to time constraints, the project's scope had to be scaled back. This meant that there was no opportunity to offer a service via Flask. Furthermore, data from the changesets could not be extracted as bad edits, as this task is also very time-consuming.

Considering that I do not have years of experience in programming, the coding part, fortunately, was not the biggest obstacle. Larger problems arose from my lack of know-how in project management for software projects.

All in all, one can look back on a satisfactory result, considering that the requirements were met.

Chapter 9. Project management

This chapter is about how the project was managed. It is a description about the developmental processes and shows the most important aspect of the project like the risk and time management.

9.1. Requirements

9.1.1. Functional Requirements

In this chapter are the details of how the functional use cases are defined.

#1 As a user, I want to call the function with a word list

The function call can be done by given any compatible word list.

#2 The function gives back all the words without any malicious word detected

In this case, the function will return an empty csv file.

#3 The function give back a warning for words ranked as bad Language

In this case, for every words that reach the criteria for being too similar to a bad word will be released. The user gets a csv file with the input word, the similar word and the similarity between them stated from the used algorithm. These words are separated by a semicolon.

#4 The format of the Input is strictly defined

The function give the possibility to receive the words to check as a csv or excel list or to give the link to an Overpass turbo query to the function.

9.1.2. Non-Functional Requirements

#NRF1 Maintainability/Modularity

Coding is done to a degree, that a user or developer can take this program for further use. Acceptance criteria: PEP 8 Style Guide for Python Code is being followed. Verification process: An automated verification in the IDE will evaluate, if the code adheres the standard.

#NFR2 Quality

The implemented profanity filter from the OSM is as a detection rate of 92% [23]. Acceptance criteria: The function must have at least one algorithm which are to 90 accurate or more. Verification process: Some tests with the collected data was done. It passed the detection rate of the already implemented filter.

9.2. Time Management

For the project planning, a Rational Unified Process was chosen. During the planning phase, it became clear that primarily data collection research would be a time-consuming process. Secondly, programming and analyzing must be evaluated to see how big the scope here can be kept, considering the building up of existing knowledge.

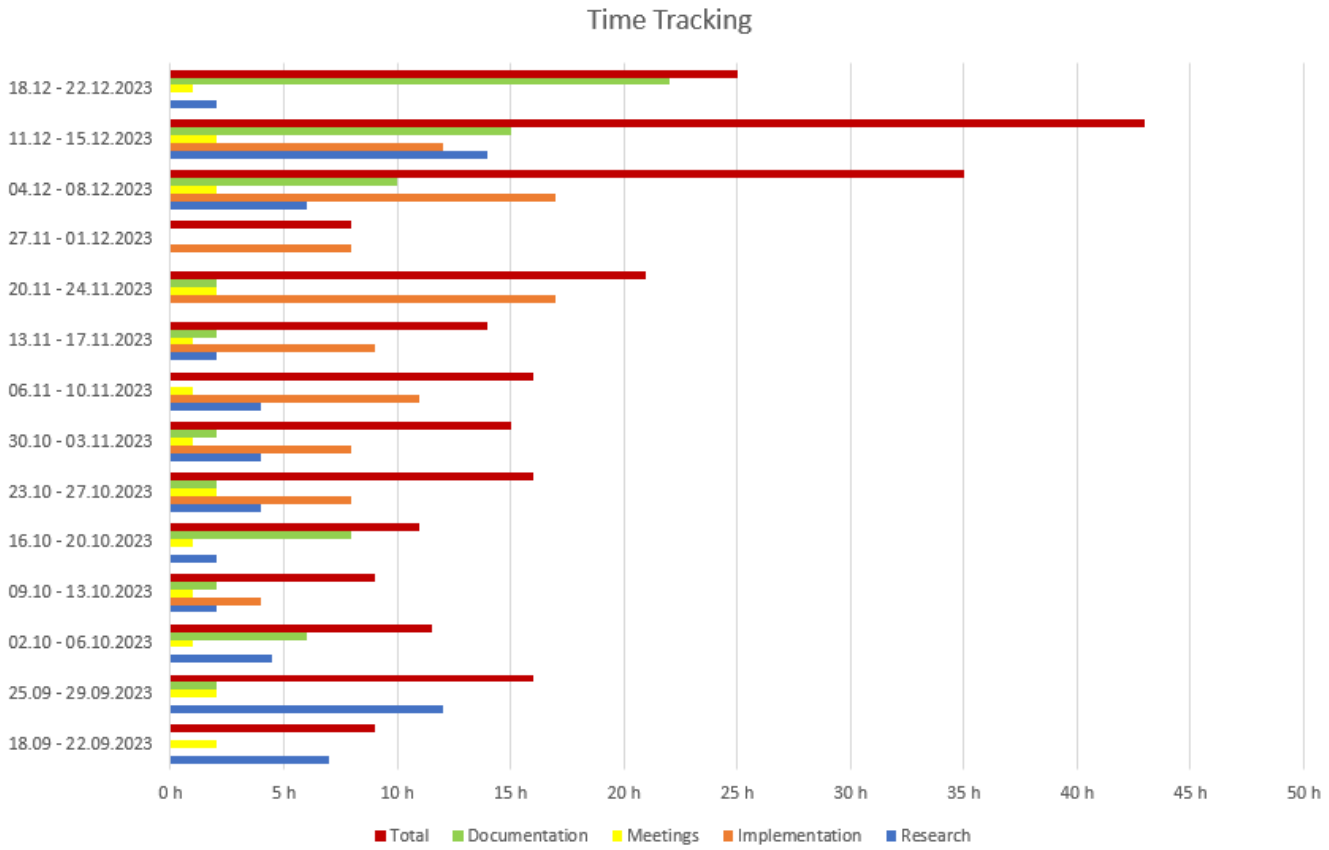

Figure 14: shows the diagram of the time tracking

Chapter 10. Installation & usage

A complete view for the installation of python and its components and usage of the intended function.

10.1. Installation

Installing python 3.14 or higher from the official webpage:

The sitePackage Openpyxl must be included in the Python interpreter as a package to get the data from the Excel files.

Modules used that must be imported: pandas csv pytest os requests from io StringIO warnings

10.1.1. Usage

There are two ways how it is intended to use the function provided. Make_csv have 5 arguments:

First argument is the provided dataset with preprocessed data in german or english which is chosen with rd.de or rd.en. It includes the bad language which will be used to compare the input strings.

Second argument is an algorithm. In this project are 4 algorithms included. If an own algorithm is chosen, there is the constraint, that the algorithm must take 1 input string and one comparison string as a parameter.

Third argument is the name of the output file. The extension of the file must be ".csv". Fourth argument is a dataset for the words which will be checked. There is the possibility to use an Excel or CSV list. The strings of the file must be in the first column starting with a title. There is prepared function for requests to catch data from the overpass API. You need to provide a link with this query:

[out:csv("name:en")]; (nwr["name"]["name:en"](46.6645,7.3114,47.8814,9.9042);

Click on the "Export" option above on the left and then right-click on the "raw data direct from Overpass API" and chose the link address.

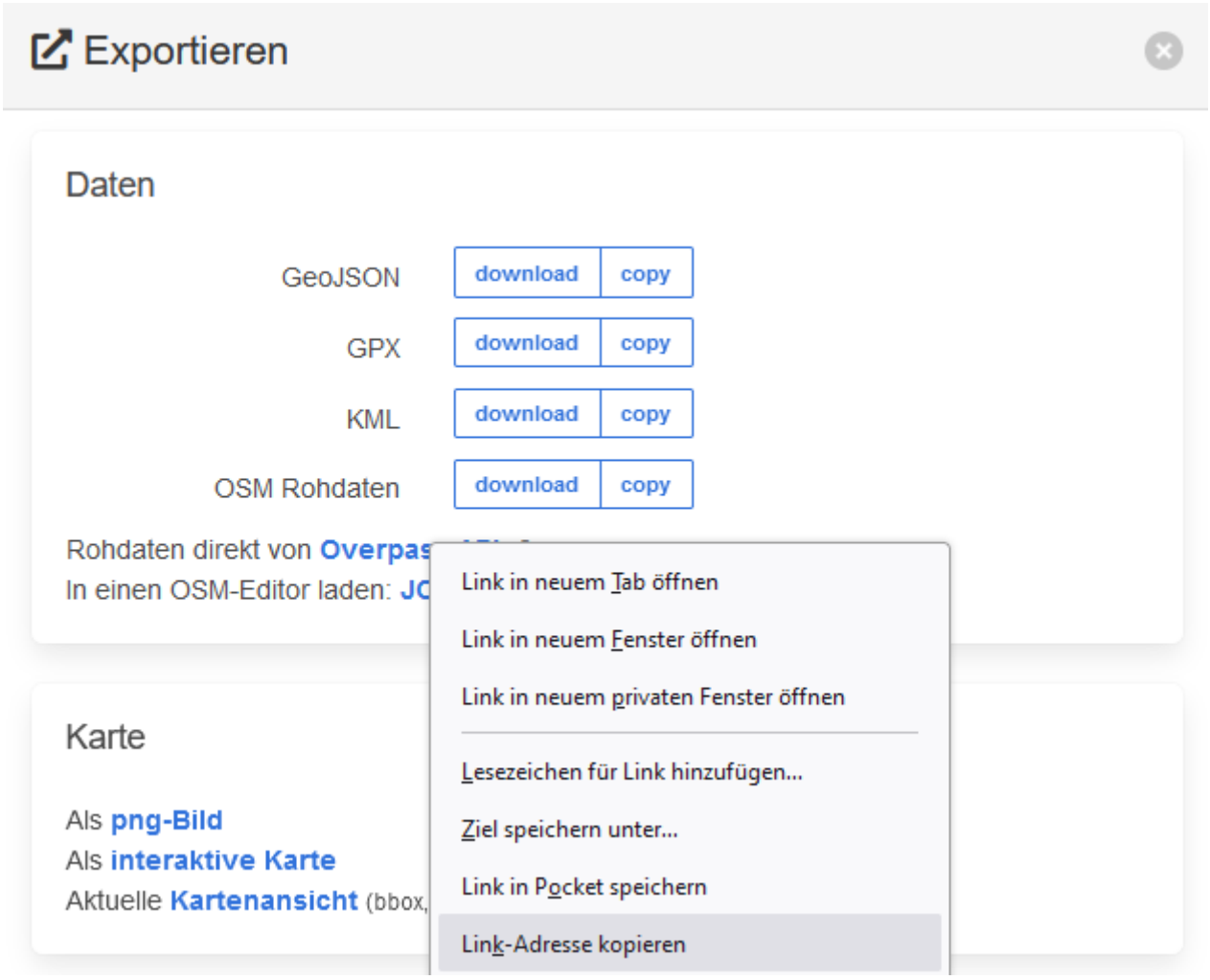

Figure 17: shows from the overpass turbo website in the window of export

The coordinates can be changed at will.

```
[out:csv("name:en")];
(
nwr["name"]["name:en"](46.6645,7.3114,47.8814,9.9042);
);
out center;
```

Figure 18: Overpass query for english names

```
[out:csv(name)];
(
nwr["name"](46.6645,7.3114,47.8814,9.9042);
);
out center;
```

Figure 19: Overpass for german names

The fifth argument is the threshold that can be chosen at will. The algorithm will distinguish good words from bad words with this value.

With that, you'll get a csv file in the working directory.

Bibliography

[1] "OpenStreetMap" [Online]. Available:
https://www.openstreetmap.org
[Accessed September 2023]

[2] "Sketchengine" [Online]. Available:
https://www.sketchengine.eu
[Accessed October 2023]

[3] "Kaggle community/blog" [Online]. Available:
https://www.kaggle.com/datasets/nicapotato
[Accessed September 2023]

[4] "OSMCha," [Online]. Available:
https://wiki.openstreetmap.org/wiki/OSMCha
[Accessed October 2023]

[5] "Changesets," [Online]. Available:
https://wiki.openstreetmap.org/wiki/List_of_Vandalism_Changesets
[Accessed September 2023]

[6] "Osm-filter"," [Online]. Available:
https://wiki.openstreetmap.org/wiki/Osmfilter
[Accessed September 2023]

[7] "Overpass turbo," [Online]. Available:
https://overpass-turbo.eu/
[Accessed October 2023]

[8] "Jaro-Winkler," [Online]. Available:
https://en.wikipedia.org/wiki/Jaro%E2%80%93Winkler_distance
[Accessed October 2023]

[9] "Levenshtein-Distance"," [Online]. Available:
https://en.wikipedia.org/wiki/Levenshtein_distance
[Accessed October 2023]

[10] "Smith-Waterman"," [Online]. Available:
Smith Waterman https://tiefenauer.github.io/blog/smith-waterman/
[Accessed November 2023]

[11] "Cosine-Distance"," [Online]. Available:
https://en.wikipedia.org/wiki/Cosine_similarity
[Accessed November 2023]

[12] "Metrics"," [Online]. Available:
https://en.wikipedia.org/wiki/F-score
[Accessed March 2023]

[13] "OSM Community," [November]. Available:
OSM Community https://community.openstreetmap.org/t/bad-langugage-example/105610
[Accessed November 2023]

[14] "OSM emergency services" [Online]. Available:
https://sosm.ch/de/community/notfallorganisationen/

[Accessed December 2023]

[15] Neis, P., Goetz, M., & Zipf, A. (2012). Towards automatic vandalism detection in OpenStreetMap ISPRS International Journal of Geo-Information, 1(3), 315-332.

[16] "OSM statistics"," [Online]. Available: http://taginfo.osm.ch/reports/database_statistics [Accessed December 2023]

[17] "Jewtropolis scandal"," [Online]. Available: https://www.openstreetmap.org/search?query=Jewett#map=14/31.3621/-96.1458 [Accessed September 2023]

[18] "OSM edit validation"," [Online]. Available: https://2018.stateofthemap.org/2018/T079-Can_we_validate_every_change_on_OSM_/ [Accessed October 2023]

[19] "wikipedia"," [Online]. Available: https://en.wikipedia.org [Accessed September 2023]

[20] "German compound splitter"," [Online]. Available: https://github.com/repodiac/german_compound_splitter/tree/master [Accessed October 2023]

[21] "Overpass documentation german," [Online]. Available: Overpass turbo de: https://dev.overpass-api.de/overpass-doc/de/ and : https://giswiki.hsr.ch/Overpass_API [Accessed October 2023]

[22] "Overpass turbo documentation en," [Online]. Available: Overpass turbo english: https://dev.overpass-api.de/overpass-doc/en/ [Accessed October 2023]

[23] "Video of the conference"," [Online]. Available: OSM conference https://www.youtube.com/watch?v=QK1M7QzaxhA [Accessed November 2023]

Not referenced in the documentation:

"Cosine distance," [Online] Available: https://stackoverflow.com/questions/3121217/cosine-similarity-of-vectors-of-different-lengths [Accessed November 2023]

"Keep Right tool", [Online]. Available: https://wiki.openstreetmap.org/wiki/Keep_Right [Accessed October 2023]

"Asciidoc," [Online]. Available:+ https://docs.asciidoctor.org/asciidoc/latest/ [Accessed September 2023]

"Chat GPT openai," [Online]. Available: Chat https://openai.com/chatgpt [Accessed October 2023]

Table Of Figures

Figures

Tables

Glossary

Name	Description
Bad Edits	These are edits that aren’t considered valuable for the development of the OSM.
Local Names	These are names used in the local area. They are important for emergency services to locate the exact position.
Changesets	If there are edits in OSM, then it will automatically be signalized as a new edit to every user. Other users will be able to tell if the edit is correct or not. This whole history of changes is changesets.
Similarity	To describe how similar a word to another word is, in this project a number between zero and one is resulting of algorithms. Zero means there is no similarity and one means they are the same word

Data analysis

cosine distance							
Tested with 0.86							
Input Word	**Bad Word**	**Similarity**	**All Input Word**	**Input Word**	**Bad Word**	**Similarity**	**All Input Words**
Rathaus Bar	Barbar	0.91287093	name	Negerweg	Neger	1	Negerweg
Platz der Erinr	Nerd	0.8660254	Bern	Schlampenpla	Schlampe	0.94280904	Schlampenpla
Platz der Skul	Nerd	0.8660254	Basel	Schwulenstras	Schwule	0.93541435	Schwulenstras
Platz der Gem	Nerd	0.8660254	Aussichtsplatt	Aidsstrasse	Aids	1	Aidsstrasse
Botschaft Arg	Hintern	0.87287156	Gurten Aussic	Hitlerweg	Hitler	1	Hitlerweg
Domus - Galle	Gelaber	0.8660254	Engelberg	Stalinplatz	Stalin	1	Stalinplatz
Jugendherber	Ungeheuer	0.89566859	Bank für Inter	Maoallee	Mao	1	Maoallee
Schweizerisch	Scheisse	0.91129565	Steffisburg	Satanstrasse	Satan	1	Satanstrasse
Verkehrspoliz	Nerd	0.8660254	Zürich	Fickgasse	Fick	1	Fickgasse
Schweine	Schweinerei	0.94311913	Graubünden/	Tittengässli	Titte	0.93541435	Tittengässli
Meerschwein	Tierchen	0.88077101	Thurgau	Pimmelweg	Pimmel	1	Pimmelweg
Bärli & Schwä	Schwächling	0.91287093	Burg Rötteln	Muschiplatz	Muschi	1	Muschiplatz
Minoretti	Marionette	0.88949918	Antikensamm	Mordweg	Mord	1	Mordweg
Jugendherber	Ungeheuer	0.89566859	Rathaus Bar	Folterplatz	Folter	1	Folterplatz
Auberge de Je	Angeber	0.88888889	Stollenrain	Sterbegasse	Streber	0.95940322	Sterbegasse
Delegation de	Nerd	0.8660254	BW-Bank	Koksweg	Koks	1	Koksweg
Schweizerisch	Schweinerei	0.89463008	Holzingers Pa	Heroinstrasse	Heroin	1	Heroinstrasse
Rote Pandas	Rosette	0.90453403	Basel-Stadt	Mordstrasse	Mord	1	Betrugsplatz
Botschaft Öst	Verschwörung	0.87287156	Platz des Feue	Bestechungsg	Bestechung	0.96896279	Mordstrasse
Gemeinschaft	Nerd	0.8660254	Platz der Erinr	Hassweg	Hass	1	Bestechungsga
Schweizerisch	Schweinerei	0.89984254	Parkhaus Zent	Angstplatz	Angst	1	Hassweg
wolkenlos - Be	Verführer	0.91465912	Garage Gut Ba	Depressionsga	Depression	0.98102294	Angstplatz
Swissmedic - S	Scheisse	0.91129565	Museggmaue	Klopapiergass	Klopapier	1	Trauerstrasse
Mark Twain G	Gesindel	0.8705715	Weggis				Depressionsga
Akademie der	Nerd	0.8660254	Platz der Skulpturen				Klopapiergass
Jugendherber	Ungeheuer	0.89566859	Urnenhain Platz Friedhof Sihlfeld				
Anderegg Gol	Angeber	0.8660254	Platz der Gemeinschaft Friedhof Sihlfeld				
			France (terres)				

Cosinus Distanz			
	Good Words		Bad Words
	0.88888889		1
	0.88883195		0.98795768
	5.6939E-05		0.01204232
False Positive	128	False Negative	2
True Positive	1166	True Negative	23
Precision	0.99828767		
Recall	0.98065601		
F1 score	0.9893933		
False Positiv Rate	0.84768212		
False Discovery Rate	0.88850772		

jaro-winkler

good words				Bad words			
Input Word	**Bad Word**	**Similarity**	**All Input Word**	**Input Word**	**Bad Word**	**Similarity**	**All Input Word**
name	Machtmensch	0.67424242	name	Negerweg	Neger	1	Negerweg
Bern	Bengel	0.75	Bern	Schlampenpla	Schlampe	0.96296296	Schlampenpla
Basel	Busen	0.73333333	Basel	Schwulenstras	Schwule	0.95833333	Schwulenstras
Aussichtsplattform (1(	Arsch	0.67407407	Aussichtsplatt	Aidsstrasse	Aids	1	Aidsstrasse
Gurten Aussichtsturm	Gutmensch	0.7962963	Gurten Aussic	Hitlerweg	Hitler	1	Hitlerweg
Engelberg	Angeber	0.84126984	Engelberg	Stalinplatz	Stalin	1	Stalinplatz
Bank für International	Bandit	0.75	Bank für Inter	Maoallee	Mao	1	Maoallee
Steffisburg	Streber	0.72294372	Steffisburg	Satanstrasse	Satan	1	Satanstrasse
Zürich	Arsch	0.7	Zürich	Fickgasse	Fick	1	Fickgasse
Graubünden/Grigioni/	Gauner	0.62698413	Graubünden/(	Tittengässli	Titte	0.94444444	Tittengässli
Thurgau	Tyrann	0.64285714	Thurgau	Pimmelweg	Pimmel	1	Pimmelweg
Burg Rötteln	Betrüger	0.70833333	Burg Rötteln	Muschiplatz	Muschi	1	Muschiplatz
Antikensammlung	Antisemit	0.7005291	Antikensamm	Mordweg	Mord	1	Mordweg
Rathaus Bar	Satan	0.67619048	Rathaus Bar	Folterplatz	Folter	1	Folterplatz
Stollenrain	Sonderling	0.80075758	Stollenrain	Sterbegasse	Streber	0.8968254	Sterbegasse
BW-Bank	Fanatiker	0.58730159	BW-Bank	Koksweg	Koks	1	Koksweg
Holzingers Pavillion	Hintern	0.73809524	Holzingers Pav	Heroinstrasse	Heroin	1	Heroinstrasse
Basel-Stadt	Bastard	0.8008658	Basel-Stadt	Betrugsplatz	Betrüger	0.7797619	Betrugsplatz
Platz des Feuers	Fratze	0.7	Platz des Feue	Mordstrasse	Mord	1	Mordstrasse
Platz der Erinnerung	Fratze	0.7	Platz der Erinn	Bestechungsg	Bestechung	0.96969697	Bestechungsg
Parkhaus Zentrum	Drecksau	0.68333333	Parkhaus Zent	Hassweg	Hass	1	Hassweg
Garage Gut Bad Ragaz	Göre	0.75	Garage Gut Ba	Angstplatz	Angst	1	Angstplatz
Museggmauer	Ausbeuter	0.73737374	Museggmauer	Trauerstrasse	Kreatur	0.78253968	Trauerstrasse
Weggis	Weib	0.75	Weggis	Depressionsga	Depression	0.96969697	Depressionsga
Platz der Skulpturen	Fratze	0.7	Platz der Skulp	Klopapiergass	Klopapier	1	Klopapiergass
Urnenhain Platz Friedl	Unsinn	0.7037037	Urnenhain Platz Friedhof Sihlfeld				
Platz der Gemeinscha	Fratze	0.7	Platz der Gemeinschaft Friedhof Sihlfeld				
Museum Rietberg	Möse	0.75	Museum Rietberg				
			Jaro Winkler				
Median		0.71111111				1	
Mittelwert		0.71067656				0.97057047	
Differenz: Median - Mittelwert		0.00043455				0.02942953	
First Try	Good Words:	0.71	Bad Words:	Suggested number		0.96	
Second Try				Suggested number		0.78	

jaro-winkler

Tested with 0.78				Tested with 0.96			
Input Word	**Bad Word**	**Similarity**	**All Input Word**	**Input Word**	**Bad Word**	**Similarity**	**All Input Word**
Gurten Aussichtsturm	Gutmensch	0.7962963	name	Negerweg	Neger	1	Negerweg
Engelberg	Angeber	0.84126984	Bern	Schlampenpla	Schlampe	0.96296296	Schlampenpla
Rathaus Bar	Barbar	0.83333333	Basel	Aidsstrasse	Aids	1	Schwulenstras
Stollenrain	Sonderling	0.80075758	Aussichtsplatt	Hitlerweg	Hitler	1	Aidsstrasse
Basel-Stadt	Bastard	0.8008658	Gurten Aussic	Stalinplatz	Stalin	1	Hitlerweg
Garage Gut Bad Ragaz	Bandit	0.83333333	Engelberg	Maoallee	Mao	1	Stalinplatz
Luzern	Luder	0.82222222	Bank für Inter	Satanstrasse	Satan	1	Maoallee
Schmitte	Schamlippe	0.78333333	Steffisburg	Fickgasse	Fick	1	Satanstrasse
Schweizerisches Hand	Schweinerei	0.82525253	Zürich	Pimmelweg	Pimmel	1	Fickgasse
Schildkröten	Scheide	0.78571429	Graubünden/(	Muschiplatz	Muschi	1	Tittengässli
Flugfeld Hausen am A	Busen	0.82222222	Thurgau	Mordweg	Mord	1	Pimmelweg
Seehund	Schlund	0.80952381	Burg Rötteln	Folterplatz	Folter	1	Muschiplatz
Schweine	Schweinerei	0.90909091	Antikensamm	Koksweg	Koks	1	Mordweg
Westafrikanische Zwe	Zwerg	0.81818182	Rathaus Bar	Heroinstrasse	Heroin	1	Folterplatz
France (terres)	Farce	0.87777778	China-Restaurant				
			Botschaft der Vereinigten Arabischen Emirate				
			France (terres)				
			Jaro Winkler				
Median		0.80952381				1	
Mittelwert		0.81500665				0.98606492	
Differenz: Median - Mittelwert		-0.00548284				0.01393508	
Final Try	Good Words	0.81	Bad Words	0.991	Suggested number		0.86

jaro-winkler							
Tested with 0.816							
Input Word	**Bad Word**	**Similarity**	**All Input Word**	**Input Word**	**Bad Word**	**Similarity**	**All Input Word**
Schweine	Schweinerei	0.90909091	name	Negerweg	Neger	1	Negerweg
Bärli & Schwänli	Schwächling	0.86742424	Bern	Schlampenpla	Schlampe	0.96296296	Schlampenpla
Botschaft Schweden	Scheide	0.86904762	Basel	Schwulenstras	Schwule	0.95833333	Schwulenstras
wolkenlos - Bergführe	Verführer	0.8962963	Aussichtsplatt	Aidsstrasse	Aids	1	Aidsstrasse
Keller AG	Killer	0.88888889	Gurten Aussic	Hitlerweg	Hitler	1	Hitlerweg
Barber Shop Maihof	Barbar	0.88888889	Engelberg	Stalinplatz	Stalin	1	Stalinplatz
Budapest Barber	Barbar	0.88888889	Bank für Inter	Maoallee	Mao	1	Maoallee
Schweizer Armee	Schweinerei	0.87205387	Steffisburg	Satanstrasse	Satan	1	Satanstrasse
Schweizer Finanzmuse	Schweinerei	0.87205387	Zürich	Fickgasse	Fick	1	Fickgasse
Coop Pronto	Porno	0.87777778	Graubünden/	Tittengässli	Titte	0.94444444	Tittengässli
Moser Küchen -Schrei	Monster	0.9047619	Thurgau	Pimmelweg	Pimmel	1	Pimmelweg
Aus der Geschichte de	Anus	0.91666667	Burg Rötteln	Muschiplatz	Muschi	1	Muschiplatz
Myer + Miller GmbH	Killer	0.88888889	Antikensamm	Mordweg	Mord	1	Mordweg
Schweizer Milchprodu	Schweinerei	0.87205387	Rathaus Bar	Folterplatz	Folter	1	Folterplatz
Anlieferung Mitte	Titte	0.86666667	Stollenrain	Sterbegasse	Streber	0.8968254	Sterbegasse
Schmiede und Metallb	Scheide	0.91071429	BW-Bank	Koksweg	Koks	1	Koksweg
Bäckerei-Konditorei A	Gangster	0.86904762	Holzingers Pa	Heroinstrasse	Heroin	1	Heroinstrasse
Schule Fokus	Schwule	0.95238095	Basel-Stadt	Mordstrasse	Mord	1	Betrugsplatz
Bus Station	Busen	0.86666667	Platz des Feue	Bestechungsg	Bestechung	0.96969697	Mordstrasse
France - Deutschland	Farce	0.87777778	Platz der Erinn	Hassweg	Hass	1	Bestechungsga
Betriebsfeuerwehr M	Bettler	0.9047619	Parkhaus Zent	Angstplatz	Angst	1	Hassweg
Ruderzentrum Luzern	Rosette	0.8968254	Garage Gut Ba	Depressionsga	Depression	0.96969697	Angstplatz
France - Schweiz/Suis	Farce	0.87777778	Museggmaue	Klopapiergass	Klopapier	1	Trauerstrasse
Basler Münster	Monster	0.9047619	Weggis				Depressionsga
Grenze Schweiz - Fran	Farce	0.87777778	Platz der Skulpturen				Klopapiergass
Schweizer Schützenm	Schweinerei	0.87205387	Urnenhain Platz Friedhof Sihlfeld				
Alte Spinnerei	Spinner	0.92592593	Platz der Gemeinschaft Friedhof Sihlfeld				
			France (terres)				

Jaro Winkler			
	Good Words		Bad Words
Median	0.87777778		1
Mittelwert	0.88263397		0.98704174
Differenz: Median - Mittelwert	-0.00485619		0.01295826
False Positive	44	False Negative	2
True Positive	1250	True Negative	23
Precision	0.99840256		
Recall	0.98193244		
F1 score	0.99009901		
False Positiv Rate	0.65671642		
False Discovery Rate	0.9632		

levenstein

good words

Input Word	Bad Word	Similarity	All Input Word
name	Fake	0.5	name
Bern	Nerd	0.5	Bern
Basel	Busen	0.6	Basel
Aussichtsplatt	Psychopath	0.33333333	Aussichtsplatt
Gurten Aussic	Hure	0.5	Gurten Aussic
Engelberg	Angeber	0.66666667	Engelberg
Bank für Inter	Bandit	0.5	Bank für Inter
Steffisburg	Streber	0.36363636	Steffisburg
Zürich	Arsch	0.5	Zürich
Graubünden/(	Gutmensch	0.25	Graubünden/(
Thurgau	Intrigant	0.33333333	Thurgau
Burg Rötteln	Hure	0.5	Burg Rötteln
Antikensamm	Antisemit	0.4	Antikensamm
Rathaus Bar	Satan	0.42857143	Rathaus Bar
Stollenrain	Stalker	0.45454545	Stollenrain
BW-Bank	Egomane	0.28571429	BW-Bank
Holzingers Pa	Hintern	0.5	Holzingers Pa
Basel-Stadt	Bastard	0.45454545	Basel-Stadt
Platz des Feue	Fratze	0.5	Platz des Feue
Platz der Erinr	Fratze	0.5	Platz der Erinr
Parkhaus Zent	Marihuana	0.44444444	Parkhaus Zent
Garage Gut Ba	Versager	0.5	Garage Gut Ba
Museggmauer	Massenmörde	0.41666667	Museggmauer
Weggis	Weib	0.5	Weggis
Platz der Skul	Fratze	0.5	Platz der Skul
Urnenhain Pla	Unsinn	0.33333333	Urnenhain Platz Friedhof Sihlfeld
Platz der Gem	Fratze	0.5	Platz der Gemeinschaft Friedhof Sihlfeld
Museum Rietl	Möse	0.5	Museum Rietberg

Levenstein

0.42857143	
0.43421344	
-0.00564201	

Good Words: 0.43

Bad Words:

Bad words

Input Word	Bad Word	Similarity	All Input Word
Negerweg	Neger	1	Negerweg
Schlampenpla	Schlampe	0.88888889	Schlampenpla
Schwulenstras	Schwule	0.875	Schwulenstras
Aidsstrasse	Aids	1	Aidsstrasse
Hitlerweg	Hitler	1	Hitlerweg
Stalinplatz	Stalin	1	Stalinplatz
Maoallee	Mao	1	Maoallee
Satanstrasse	Satan	1	Satanstrasse
Fickgasse	Fick	1	Fickgasse
Tittengässli	Titte	0.83333333	Tittengässli
Pimmelweg	Pimmel	1	Pimmelweg
Muschiplatz	Muschi	1	Muschiplatz
Mordweg	Mord	1	Mordweg
Folterplatz	Folter	1	Folterplatz
Sterbegasse	Streber	0.57142857	Sterbegasse
Koksweg	Koks	1	Koksweg
Heroinstrasse	Heroin	1	Heroinstrasse
Betrugsplatz	Betrüger	0.625	Betrugsplatz
Mordstrasse	Mord	1	Mordstrasse
Bestechungsg	Bestechung	0.90909091	Bestechungsg
Hassweg	Hass	1	Hassweg
Angstplatz	Angst	1	Angstplatz
Trauerstrasse	Versager	0.5	Trauerstrasse
Depressionsga	Depression	0.90909091	Depressionsga
Klopapiergass	Klopapier	1	Klopapiergass

Levenstein

1	
0.9244733	
0.0755267	

Suggested number 0.9

Suggested number 0.83

levenstein

Tested with 0.83

Input Word	Bad Word	Similarity	All Input Word
Keller AG	Killer	0.83333333	name
Barber Shop N	Barbar	0.83333333	Bern
Budapest Bark	Barbar	0.83333333	Basel
Moser Kücher	Schweinerei	0.83333333	Aussichtsplatt
Myer + Miller	Killer	0.83333333	Gurten Aussic
Schule Fokus	Schwule	0.85714286	Engelberg
Betriebsfeuer	Bettler	0.85714286	Bank für Inter
Basler Münste	Monster	0.85714286	Steffisburg
			Zürich
			Graubünden/(
			Thurgau
			Burg Rötteln
			Antikensamm
			Rathaus Bar
			China-Restaurant
			Botschaft der Vereinigten Arabischen Emirate
			France (terres)

Levenstein

0.83333333	
0.8422619	
-0.00892857	

Tested with 0.9

Input Word	Bad Word	Similarity	All Input Word
Negerweg	Neger	1	Negerweg
Aidsstrasse	Aids	1	Schlampenpla
Hitlerweg	Hitler	1	Schwulenstras
Stalinplatz	Stalin	1	Aidsstrasse
Maoallee	Mao	1	Hitlerweg
Satanstrasse	Satan	1	Stalinplatz
Fickgasse	Fick	1	Maoallee
Pimmelweg	Pimmel	1	Satanstrasse
Muschiplatz	Muschi	1	Fickgasse
Mordweg	Mord	1	Tittengässli
Folterplatz	Folter	1	Pimmelweg
Koksweg	Koks	1	Muschiplatz
Heroinstrasse	Heroin	1	Mordweg
Mordstrasse	Mord	1	Folterplatz

Good Words	0.838	Bad Words	0.986	Suggested number	0.86

levenstein							
Tested with 0.86							
Input Word	**Bad Word**	**Similarity**	**All Input Word**	**Input Word**	**Bad Word**	**Similarity**	**All Input Word**
			name	Negerweg	Neger	1	Negerweg
tz			Bern	Schlampenpla	Schlampe	0.88888889	Schlampenpla
sse			Basel	Schwulenstras	Schwule	0.875	Schwulenstras
			Aussichtsplatt	Aidsstrasse	Aids	1	Aidsstrasse
			Gurten Aussic	Hitlerweg	Hitler	1	Hitlerweg
			Engelberg	Stalinplatz	Stalin	1	Stalinplatz
			Bank für Inter	Maoallee	Mao	1	Maoallee
			Steffisburg	Satanstrasse	Satan	1	Satanstrasse
			Zürich	Fickgasse	Fick	1	Fickgasse
			Graubünden/	Pimmelweg	Pimmel	1	Tittengässli
			Thurgau	Muschiplatz	Muschi	1	Pimmelweg
			Burg Rötteln	Mordweg	Mord	1	Muschiplatz
			Antikensamm	Folterplatz	Folter	1	Mordweg
			Rathaus Bar	Koksweg	Koks	1	Folterplatz
			Stollenrain	Heroinstrasse	Heroin	1	Sterbegasse
			BW-Bank	Mordstrasse	Mord	1	Koksweg
			Holzingers Pa	Bestechungsg	Bestechung	0.90909091	Heroinstrasse
			Basel-Stadt	Hassweg	Hass	1	Betrugsplatz
			Platz des Feue	Angstplatz	Angst	1	Mordstrasse
asse			Platz der Erinn	Depressionsga	Depression	0.90909091	Bestechungsg
			Parkhaus Zent	Klopapiergass	Klopapier	1	Hassweg
			Garage Gut Bad Ragaz				Angstplatz
			Museggmauer				Trauerstrasse
asse			Weggis				Depressionsga
e			Platz der Skulpturen				Klopapiergass
			Urnenhain Platz Friedhof Sihlfeld				
			Platz der Gemeinschaft Friedhof Sihlfeld				
			France (terres)				

Levenstein			
Good Words		Bad Words	
#ZAHL!		1	
#DIV/0!		0.98009861	
#ZAHL!		0.01990139	
False Positive	0	False Negative	4
True Positive	1294	True Negative	21
Precision	0.99691834		
Recall	0.98403042		
F1 score	0.99043245		
False Positiv Rate	0		
False Discovery Rate	0.99690881		

smith-waterman

good words				Bad words			
Input Word	**Bad Word**	**Similarity**	**All Input Word**	**Input Word**	**Bad Word**	**Similarity**	**All Input Word**
name	Fake	0.75	name	Negerweg	Neger	1	Negerweg
Bern	Bengel	0.8	Bern	Schlampenpla	Schlampe	0.94117647	Schlampenpla
Basel	Busen	0.8	Basel	Schwulenstras	Schnauze	1	Schwulenstras
Aussichtsplatt	Psychopath	0.71428571	Aussichtsplatt	Aidsstrasse	Aids	1	Aidsstrasse
Gurten Aussic	Göre	1	Gurten Aussic	Hitlerweg	Hetzer	1	Hitlerweg
Engelberg	Langweiler	0.94736842	Engelberg	Stalinplatz	Stalin	1	Stalinplatz
Bank für Inter	Fake	0.75	Bank für Inter	Maoallee	Mao	1	Maoallee
Steffisburg	Kriegstreiber	0.83333333	Steffisburg	Satanstrasse	Satan	1	Satanstrasse
Zürich	Arsch	0.72727273	Zürich	Fickgasse	Fick	1	Fickgasse
Graubünden/(	Sonderling	0.57894737	Graubünden/(	Tittengässli	Flittchen	0.93333333	Tittengässli
Thurgau	Kreatur	0.71428571	Thurgau	Pimmelweg	Pimmel	1	Pimmelweg
Burg Rötteln	Unfug	0.66666667	Burg Rötteln	Muschiplatz	Muschi	1	Muschiplatz
Antikensamm	Fiesling	0.8	Antikensamm	Mordweg	Mord	1	Mordweg
Rathaus Bar	Bastard	0.71428571	Rathaus Bar	Folterplatz	Fotzen	1	Folterplatz
Stollenrain	Störenfried	1	Stollenrain	Sterbegasse	Streber	0.8	Sterbegasse
BW-Bank	Fake	0.54545455	BW-Bank	Koksweg	Koks	1	Koksweg
Holzingers Pa	Halunke	0.94117647	Holzingers Pa	Heroinstrasse	Heroin	1	Heroinstrasse
Basel-Stadt	Bastard	0.8	Basel-Stadt	Betrugsplatz	Biest	0.83333333	Betrugsplatz
Platz des Feue	Chaot	0.6	Platz des Feue	Mordstrasse	Mord	1	Mordstrasse
Platz der Erinr	Chaot	0.6	Platz der Erinr	Bestechungsg	Bestechung	0.95238095	Bestechungsg
Parkhaus Zent	Marihuana	0.70588235	Parkhaus Zent	Hassweg	Hass	1	Hassweg
Garage Gut Ba	Versager	0.71428571	Garage Gut Ba	Angstplatz	Angst	1	Angstplatz
Museggmauer	Massenmörde	0.86956522	Museggmauer	Trauerstrasse	Versager	0.85714286	Trauerstrasse
Weggis	Feigling	0.71428571	Weggis	Depressionsga	Depression	0.95238095	Depressionsga
Platz der Skul	Chaot	0.6	Platz der Skul	Klopapiergass	Klopapier	1	Klopapiergass
Urnenhain Pla	Ungeheuer	0.77777778	Urnenhain Platz Friedhof Sihlfeld				
Platz der Gem	Chaot	0.6	Platz der Gemeinschaft Friedhof Sihlfeld				
Museum Riet	Möse	0.8	Museum Rietberg				
			Smith waterman				
		0.77777778				1	
		0.75174895				0.97078992	
		0.02602882				0.02921008	
Good Words:		0.7	Bad Words:	Suggested number		0.96	
				Suggested number		0.83	

smith-waterman

Tested with 0.82				Tested with 0.96			
Input Word	**Bad Word**	**Similarity**	**All Input Word**	**Input Word**	**Bad Word**	**Similarity**	**All Input Word**
Aussichtsplatt	Stalker	0.88888889	name	Negerweg	Neger	1	Negerweg
Gurten Aussic	Göre	1	Bern	Schwulenstras	Schnauze	1	Schlampenpla
Engelberg	Langweiler	0.94736842	Basel	Aidsstrasse	Aids	1	Schwulenstras
Bank für Inter	Ignorant	0.86956522	Aussichtsplatt	Hitlerweg	Hetzer	1	Aidsstrasse
Steffisburg	Kriegstreiber	0.83333333	Gurten Aussic	Stalinplatz	Stalin	1	Hitlerweg
Burg Rötteln	Flittchen	0.875	Engelberg	Maoallee	Mao	1	Stalinplatz
Stollenrain	Störenfried	1	Bank für Inter	Satanstrasse	Satan	1	Maoallee
Holzingers Pa	Halunke	0.94117647	Steffisburg	Fickgasse	Fick	1	Satanstrasse
Platz des Feue	Aids	0.85714286	Zürich	Pimmelweg	Pimmel	1	Fickgasse
Garage Gut Ba	Bandit	0.88888889	Graubünden/(	Muschiplatz	Muschi	1	Tittengässli
Museggmauer	Massenmörde	0.86956522	Thurgau	Mordweg	Mord	1	Pimmelweg
Platz der Skul	Vollpfosten	0.95238095	Burg Rötteln	Folterplatz	Fotzen	1	Muschiplatz
Urnenhain Pla	Eichel	1	Antikensamm	Koksweg	Koks	1	Mordweg
Platz der Gem	Eichel	1	Rathaus Bar	Heroinstrasse	Heroin	1	Folterplatz
Schliessfächer	Machtmensch	0.88	China-Restaurant				
Platz des Tros	Aids	0.85714286	Botschaft der Vereinigten Arabischen Emirate				
			France (terres)				
			Smith waterman				
		0.90909091				1	
		0.91441339				0.98737495	
		-0.00532248				0.01262505	
	Good Words	0.911	Bad Words	0.992	Suggested number		0.94

smith-waterman							
Tested with 0.915							
Input Word	**Bad Word**	**Similarity**	**All Input Word**	**Input Word**	**Bad Word**	**Similarity**	**All Input Word**
Gurten Aussic	Göre	1	name	Negerweg	Neger	1	Negerweg
Engelberg	Langweiler	0.94736842	Bern	Schlampenpla	Schlampe	0.94117647	Schlampenpla
Stollenrain	Störenfried	1	Basel	Schwulenstras	Schnauze	1	Schwulenstras
Holzingers Pa	Halunke	0.94117647	Aussichtsplatt	Aidsstrasse	Aids	1	Aidsstrasse
Platz des Feue	Heuchler	1	Gurten Aussic	Hitlerweg	Hetzer	1	Hitlerweg
Platz der Skul	Vollpfosten	0.95238095	Engelberg	Stalinplatz	Stalin	1	Stalinplatz
Urnenhain Pla	Eichel	1	Bank für Inter	Maoallee	Mao	1	Maoallee
Platz der Gem	Eichel	1	Steffisburg	Satanstrasse	Satan	1	Satanstrasse
Botschaft Arg	Psychopath	0.94736842	Zürich	Fickgasse	Fick	1	Fickgasse
Botschaft Mal	Psychopath	0.94736842	Graubünden/	Pimmelweg	Pimmel	1	Tittengässli
Botschaft Por	Psychopath	0.94736842	Thurgau	Muschiplatz	Muschi	1	Pimmelweg
Botschaft Me	Psychopath	0.94736842	Burg Rötteln	Mordweg	Mord	1	Muschiplatz
Tanzschule Nc	Taugenichts	0.95238095	Antikensamm	Folterplatz	Fotzen	1	Mordweg
Restaurierung	Kriegstreiber	1	Rathaus Bar	Koksweg	Koks	1	Folterplatz
Botschaft Chil	Psychopath	0.94736842	Stollenrain	Heroinstrasse	Heroin	1	Sterbegasse
Schmitte	Schwätzer	0.94117647	BW-Bank	Mordstrasse	Mord	1	Koksweg
Botschaft Gha	Psychopath	0.94736842	Holzingers Pa	Bestechungsg	Bestechung	0.95238095	Heroinstrasse
Fischotter	Mistkerl	1	Basel-Stadt	Hassweg	Hass	1	Betrugsplatz
Landtag des F	Heuchelei	1	Platz des Feue	Angstplatz	Angst	1	Mordstrasse
Kindergarten	Verrückte	0.95238095	Platz der Erinn	Depressionsga	Depression	0.95238095	Bestechungsg
Flugfeld Haus	Hure	1	Parkhaus Zent	Klopapiergass	Klopapier	1	Hassweg
Meltingen	Alien	1	Garage Gut Bad Ragaz				Angstplatz
Schmutzgeier	Schnauze	1	Museggmauer				Trauerstrasse
Grenchen	Hurensohn	0.94117647	Weggis				Depressionsga
Bärli & Schwä	Schwächling	0.94736842	Platz der Skulpturen				Klopapiergass
Spice India	Intrigant	1	Urnenhain Platz Friedhof Sihlfeld				
Minoretti	Hinterteil	0.94736842	Platz der Gemeinschaft Friedhof Sihlfeld France (terres)				

Smith waterman			
	Good Words		Bad Words
	1		1
	0.97665048		0.99266373
	0.02334952		0.00733627
False Positive	309	False Negative	4
True Positive	985	True Negative	21
Precision	0.99595551		
Recall	0.97912525		
F1 score	0.98746867		
False Positiv Rate	0.93636364		
False Discovery Rate	0.6822335		

cosine distance

good words				Bad words			
Input Word	**Bad Word**	**Similarity**	**All Input Word**	**Input Word**	**Bad Word**	**Similarity**	**All Input Word**
name	Egomane	0.75592895	name	Negerweg	Neger	1	Negerweg
Bern	Penner	0.79056942	Bern	Schlampenpla	Schlampe	0.94280904	Schlampenpla
Basel	Selbstdarstelle	0.69026849	Basel	Schwulenstras	Schwule	0.93541435	Schwulenstras
Aussichtsplatt	Faschist	0.75055535	Aussichtsplatt	Aidsstrasse	Aids	1	Aidsstrasse
Gurten Aussic	Gauner	0.83333333	Gurten Aussic	Hitlerweg	Hitler	1	Hitlerweg
Engelberg	Angeber	0.83205029	Engelberg	Stalinplatz	Stalin	1	Stalinplatz
Bank für Inter	Bandit	0.61237244	Bank für Inter	Maoallee	Mao	1	Maoallee
Steffisburg	Säufer	0.67936622	Steffisburg	Satanstrasse	Satan	1	Satanstrasse
Zürich	Wichser	0.6172134	Zürich	Fickgasse	Fick	1	Fickgasse
Graubünden/(	Wahnsinnige	0.69551861	Graubünden/(	Tittengässli	Titte	0.93541435	Tittengässli
Thurgau	Taugenichts	0.60302269	Thurgau	Pimmelweg	Pimmel	1	Pimmelweg
Burg Rötteln	Draufgänger	0.64549722	Burg Rötteln	Muschiplatz	Muschi	1	Muschiplatz
Antikensamm	Wahnsinnige	0.71754731	Antikensamm	Mordweg	Mord	1	Mordweg
Rathaus Bar	Quatsch	0.75592895	Rathaus Bar	Folterplatz	Folter	1	Folterplatz
Stollenrain	Stalin	0.84327404	Stollenrain	Sterbegasse	Streber	0.95940322	Sterbegasse
BW-Bank	Bandit	0.54433105	BW-Bank	Koksweg	Koks	1	Koksweg
Holzingers Pa	Heroin	0.77459667	Holzingers Pa	Heroinstrasse	Heroin	1	Heroinstrasse
Basel-Stadt	Bastard	0.77459667	Basel-Stadt	Betrugsplatz	Betrüger	0.76376262	Betrugsplatz
Platz des Feue	Maulfotze	0.59628479	Platz des Feue	Mordstrasse	Mord	1	Mordstrasse
Platz der Erinr	Maulfotze	0.59628479	Platz der Erinr	Bestechungsg	Bestechung	0.96896279	Bestechungsga
Parkhaus Zent	Marihuana	0.73484692	Parkhaus Zent	Hassweg	Hass	1	Hassweg
Garage Gut Ba	Gangster	0.75	Garage Gut Ba	Angstplatz	Angst	1	Angstplatz
Museggmauer	Versager	0.70014004	Museggmauer	Trauerstrasse	Kreatur	0.82495791	Trauerstrasse
Weggis	Feigling	0.71443451	Weggis	Depressionsga	Depression	0.98102294	Depressionsga
Platz der Skulp	Maulfotze	0.59628479	Platz der Skulp	Klopapiergass	Klopapier	1	Klopapiergass
Urnenhain Pla	Wahnsinnige	0.82928843	Urnenhain Platz Friedhof Sihlfeld				
Platz der Gem	Maulfotze	0.59628479	Platz der Gemeinschaft Friedhof Sihlfeld				
Museum Rietb	Ausbeuter	0.68640647	Museum Rietberg				
			Cosinus Distanz				
		0.73029674				1	
		0.71849757				0.97246989	
		0.01179917				0.02753011	
Good Words:		0.72	Bad Words:	Suggested number		0.96	
				Suggested number		0.82	

cosine distance

Tested with 0.82				Tested with 0.96			
Input Word	**Bad Word**	**Similarity**	**All Input Word**	**Input Word**	**Bad Word**	**Similarity**	**All Input Word**
Gurten Aussic	Gauner	0.83333333	name	Negerweg	Neger	1	Negerweg
Engelberg	Angeber	0.83205029	Bern	Aidsstrasse	Aids	1	Schlampenpla
Bank für Inter	Ignorant	0.82572282	Basel	Hitlerweg	Hitler	1	Schwulenstras
Rathaus Bar	Barbar	0.91287093	Aussichtsplatt	Stalinplatz	Stalin	1	Aidsstrasse
Stollenrain	Stalin	0.84327404	Gurten Aussic	Maoallee	Mao	1	Hitlerweg
Platz der Erinr	Nerd	0.8660254	Engelberg	Satanstrasse	Satan	1	Stalinplatz
Platz der Skulp	Nerd	0.8660254	Bank für Inter	Fickgasse	Fick	1	Maoallee
Urnenhain Pla	Wahnsinnige	0.82928843	Steffisburg	Pimmelweg	Pimmel	1	Satanstrasse
Platz der Gem	Nerd	0.8660254	Zürich	Muschiplatz	Muschi	1	Fickgasse
Museum Rietb	Kriegstreiber	0.85201287	Graubünden/(	Mordweg	Mord	1	Tittengässli
Botschaft Arg	Hintern	0.87287156	Thurgau	Folterplatz	Folter	1	Pimmelweg
Zabele Restau	Gelaber	0.82495791	Burg Rötteln	Koksweg	Koks	1	Muschiplatz
Restaurierung	Kriegstreiber	0.8247861	Antikensamm	Heroinstrasse	Heroin	1	Mordweg
Domus - Galle	Gelaber	0.8660254	Rathaus Bar	Mordstrasse	Mord	1	Folterplatz
			China-Restaurant				
			Botschaft der Vereinigten Arabischen Emirate				
			France (terres)				
			Cosinus Distanz				
		0.84527852				1	
		0.85681914				0.98832489	
		-0.0115406				0.01167511	
Good Words		0.85	Bad Words	0.993	Suggested number		0.86

cosine distance							
Tested with 0.86							
Input Word	**Bad Word**	**Similarity**	**All Input Word**	**Input Word**	**Bad Word**	**Similarity**	**All Input Words**
Rathaus Bar	Barbar	0.91287093	name	Negerweg	Neger	1	Negerweg
Platz der Erinr	Nerd	0.8660254	Bern	Schlampenpla	Schlampe	0.94280904	Schlampenpla
Platz der Skul	Nerd	0.8660254	Basel	Schwulenstras	Schwule	0.93541435	Schwulenstras
Platz der Gem	Nerd	0.8660254	Aussichtsplatt	Aidsstrasse	Aids	1	Aidsstrasse
Botschaft Arg	Hintern	0.87287156	Gurten Aussic	Hitlerweg	Hitler	1	Hitlerweg
Domus - Galle	Gelaber	0.8660254	Engelberg	Stalinplatz	Stalin	1	Stalinplatz
Jugendherber	Ungeheuer	0.89566859	Bank für Inter	Maoallee	Mao	1	Maoallee
Schweizerisch	Scheisse	0.91129565	Steffisburg	Satanstrasse	Satan	1	Satanstrasse
Verkehrspoliz	Nerd	0.8660254	Zürich	Fickgasse	Fick	1	Fickgasse
Schweine	Schweinerei	0.94311913	Graubünden/	Tittengässli	Titte	0.93541435	Tittengässli
Meerschwein	Tierchen	0.88077101	Thurgau	Pimmelweg	Pimmel	1	Pimmelweg
Bärli & Schwä	Schwächling	0.91287093	Burg Rötteln	Muschiplatz	Muschi	1	Muschiplatz
Minoretti	Marionette	0.88949918	Antikensamm	Mordweg	Mord	1	Mordweg
Jugendherber	Ungeheuer	0.89566859	Rathaus Bar	Folterplatz	Folter	1	Folterplatz
Auberge de Je	Angeber	0.88888889	Stollenrain	Sterbegasse	Streber	0.95940322	Sterbegasse
Delegation de	Nerd	0.8660254	BW-Bank	Koksweg	Koks	1	Koksweg
Schweizerisch	Schweinerei	0.89463008	Holzingers Pa	Heroinstrasse	Heroin	1	Heroinstrasse
Rote Pandas	Rosette	0.90453403	Basel-Stadt	Mordstrasse	Mord	1	Betrugsplatz
Botschaft Öst	Verschwörung	0.87287156	Platz des Feue	Bestechungsg	Bestechung	0.96896279	Mordstrasse
Gemeinschaft	Nerd	0.8660254	Platz der Erinr	Hassweg	Hass	1	Bestechungsga
Schweizerisch	Schweinerei	0.89984254	Parkhaus Zent	Angstplatz	Angst	1	Hassweg
wolkenlos - Be	Verführer	0.91465912	Garage Gut Ba	Depressionsga	Depression	0.98102294	Angstplatz
Swissmedic - S	Scheisse	0.91129565	Museggmaue	Klopapiergass	Klopapier	1	Trauerstrasse
Mark Twain G	Gesindel	0.8705715	Weggis				Depressionsga
Akademie der	Nerd	0.8660254	Platz der Skulpturen				Klopapiergass
Jugendherber	Ungeheuer	0.89566859	Urnenhain Platz Friedhof Sihlfeld				
Anderegg Gol	Angeber	0.8660254	Platz der Gemeinschaft Friedhof Sihlfeld				
			France (terres)				

Cosinus Distanz			
	Good Words		Bad Words
	0.88888889		1
	0.88883195		0.98795768
	5.6939E-05		0.01204232
False Positive	128	False Negative	2
True Positive	1166	True Negative	23
Precision	0.99828767		
Recall	0.98065601		
F1 score	0.9893933		
False Positiv Rate	0.84768212		
False Discovery Rate	0.88850772		

Jaro Winkler			Levenstein		Smith Waterman		Cosinus Distanz	
	Good Words	Bad Words	Good Words	Bad Words	Good Words	Bad Words	Good Words	Bad Words
Median	0.87777778	1	0	1	1	1	0.888888889	1
Mittelwert	0.88263397	0.98704174	0	0.98009861	0.97665048	0.99266373	0.88883195	0.98795768
Differenz: Median - Mittelwe	-0.00485619	0.01295826	0	0.01990139	0.02334952	0.00733627	5.69385E-05	0.01204232
False Positiv:	44		0		309		128	
True Positiv:	1250		1294		985		1166	
False Negativ	2		4		4		2	
True Negativ	23		21		21		23	
Precision	0.99840256		0.99691834		0.99595551		0.998287671	
Recall	0.98193244		0.98403042		0.97912525		0.980656013	
F1 score	0.99009901		0.99043245		0.98746867		0.989393297	
False Positiv Rate	0.65671642		0		0.93636364		0.847682119	
False Discovery Rate	0.9632		0.99690881		0.6822335		0.888507719	
True Positve Rate	0.92		0.84		0.84		0.92	