# Enhanced Address Search with Spelling Variants

Konstantin Clemens
*Technische Universität Berlin, Service-centric Networking, Germany*

Keywords: Geocoding, Postal Address Search, Spelling Variant, Spelling Error, Document Search.

Abstract: The process of resolving names of spatial entities like postal addresses or administrative areas into their whereabouts is called *geocoding*. It is an error-prone process for multiple reasons: Names of postal address elements like cities, streets, or districts are often reused for historical reasons; structures of postal addresses are only coherent within countries or regions - around the globe addresses are not structured in a canonical way; human users might not adhere even to locally common format for specifying addresses; also, humans often introduce spelling mistakes when referring to a location.
In this paper, a log of address searches from human users is used to model user behavior with regards to spelling mistakes. This model is used to generate spelling variants of address tokens which are indexed in addition to the proper spelling. Experiments show that augmenting the index of a geocoder with spelling variants is a valuable approach to handling queries with misspelled tokens. It enables the system to serve more such queries correctly as compared to a geocoding system supporting edit distances: While this way the recall of such a system is improved, its precision remains on par at the same time.

## 1 INTRODUCTION

Nowadays digital maps and digital processing of location information are popularly used. Besides various applications for automated processing of location data, like (Can et al., 2005), (Sengar et al., 2007), (Borkar et al., 2000), or (Srihari, 1993), users rely on computers to navigate through an unknown area or to store, retrieve, and display location information. Withal, internally, computers reference locations through a coordinate system such as WGS84 latitude and longitude coordinates (National Imagery and Mapping Agency, 2004). Human users, on the other hand, refer to locations by addresses or common names. The process of mapping such names or addresses to their location on a coordinate system is called *geocoding*.

There are two aspects to this error-prone process (Fitzke and Atkinson, 2006), (Ge et al., 2005), (Goldberg et al., 2007), (Drummond, 1995): First, the geocoding system needs to parse the user query and derive the query intent, i.e., the system needs to understand which address entity the query refers to. Then, the system needs to look up the coordinates of the entity the query was referring to and return it as a result. Already the first step is a non-trivial task, especially when considering the human factor: Some address elements are often misspelled or abbreviated by users in a non-standard way. Also, while postal addresses seem structured and like they adhere to a well-defined format, (Clemens, 2013) shows that each format only holds within a specific region. Considering addresses from all over the world, address formats often contradict to each other, so that there is no pattern that all queries would fit in. In addition to that, like with spelling errors, human users may not adhere to a format, leaving names of address elements out or specifying them in an unexpected order. Such incomplete or missorted queries are often ambiguous, as the same names are reused for different and often times unrelated address elements. Various algorithms are employed to mitigate these issues. Even with the best algorithms at hand, however, a geocoding service can only be as good as the data it builds upon, as understanding the query intent is not leading to a good geocoding result if, e.g., there is no data to return.

Many on-line geocoding services like those offered by Google (Google, 2017), Yandex (Yandex, 2017), Yahoo! (Yahoo!, 2017), HERE (HERE, 2017), or OpenStreetMap (OpenStreetMap Foundation, 2017b) are easily accessible by the end user. Because most of these systems are proprietary solutions, they neither reveal the data nor the algorithms used. This makes it hard to compare distinct aspects of such services. An exception to that is OpenStreetMap: The crowd-sourced data is publicly available for everyone. Open-source projects like Nominatim



Clemens, K.
Enhanced Address Search with Spelling Variants.
DOI: 10.5220/0006646100280035
In *Proceedings of the 4th International Conference on Geographical Information Systems Theory, Applications and Management (GISTAM 2018)*, pages 28-35
ISBN: 978-989-758-294-3

(OpenStreetMap Foundation, 2017a) provide geocoding services on top of that. In this paper, data from OpenStreetMap is used to create a geocoding service that is capable of deriving the user intent from a query, even if it contains spelling errors or is stated in a non-standard format. Nominatim - the reference geocoder for OpenStreetMap data - is used as one of the baselines to compare with. Thereby, the *recall* of a geocoding system is the ratio of successful responses containing the result queried for, while as the *precision* describes the ratio of responses not containing different and therefore wrong results. For ambiguous queries most geocoding systems return responses with multiple results. Obviously, at most one result can be the one queried for, while all other results can only be wrong. Therefore, such responses can be regarded as either successfully served and increasing the recall, or as failures reducing precision. Because this paper aims at increasing the recall by reducing the ambiguity of queries, each response with more than one result is counted as non-successful, affecting the precision metric of the respective geocoder negatively.

In this paper a novel approach is suggested to increase the recall of a geocoder. The idea is to make the system capable of supporting specific, most commonly made spelling errors. Usually, this is achieved by allowing edit distances between tokens of the query and the address. That, however, inherently increases the ambiguity of queries and leads to a lower precision of the system: More responses contain results that queries did not refer to. The suggested approach aims to avoid that by only allowing specific spelling variants that are made often, while avoiding spelling variants that are not made at all - edit distances lack this differentiation.

For that, from a log of real user queries the most common spelling mistakes users make are derived. These spelling variants are indexed in addition to the correctly spelled address tokens. Variants of geocoding systems created this way are evaluated with regard to their precision and recall metrics, and compared to a similar system supporting edit distances, as well as Nominatim. In (Clemens, 2015a) and (Clemens, 2015b), similar measurements have shown that TF/IDF (Salton and Yang, 1973) (Salton et al., 1975) or BM25f (Robertson et al., 2004) based document search engines like Elasticsearch (Elastic, 2017) handle incomplete or shuffled queries much better than Nominatim. This paper is a continuation of that work. It adds to both the indexing mechanism proposed in (Clemens, 2015a) and (Clemens, 2015b) as well as the way the system performance is measured.

Work on comparing geocoding services has been undertaken in, e.g., (Yang et al., 2004), (Davis and Fonseca, 2007), (Roongpiboonsopit and Karimi, 2010), or (Duncan et al., 2011). Mostly, such works focus on the recall aspect of a geocoder: Only how often a system can find the right result is compared. Also, other evaluations of geocoding systems treat every system as a black box. Thus, a system can be algorithmically strong, but perform poorly in a measurement because it is lacking data. Vice versa, a system can look better than others just because of great data coverage, despite being algorithmically poor. In this paper, the algorithmic aspect is evaluated in isolation, as all systems are set up with the same data. Also, a different way of measuring the geocoders performance is proposed: Based on real user queries a statistical model is created which is used to generate erroneous, user-like queries out of any given valid address. This approach allows to measure a system on a much greater number of addresses.

Another approach to the geocoding problem is to find an address schema that is easy to use and standardized in a non-contradicting way. While current schemata of postal addresses are maintained by the UPU (Universal Postal Union, 2017), approaches like (what3words, 2017), (Coetzee et al., 2008), (Mayrhofer and Spanring, 2010), (Fang et al., 2010), or (geo poet, 2017) are suggesting standardized or entirely alternative address schemata. (Clemens, 2016) shows that such address schemata are beneficial in some scenarios, though they are far from being adopted into everyday use.

In the next section, the steps to set up such geocoding systems are described. Afterwards, in Section 3 the undertaken measurements are described in detail. Next, in Section 4, the observed results are discussed and interpreted. Finally, in the last section, the conclusions are summarized and further work is discussed.

## 2 SETTING UP A GEOCODER

The experiment is conducted on the OpenStreetMap data set for Europe. This data set is not collected with a specific application in mind. For many use cases, it needs to be preprocessed from its raw format before it can be consumed. As in (Clemens, 2015a) and (Clemens, 2015b), the process for preprocessing OpenStreetMap data built into Nominatim has been used. Though a long-lasting task, reusing this process ensures all systems are set up with exactly the same data, thereby enabling the comparability of the algorithmic part of those systems. Thus, first, Nomi-

```
ID: 20
TEXT: Ernst Reuter Platz, 10587 Berlin
LATLON: 52.5127183,13.3217624
HOUSE NUMBERS:
 - 7
    ID:200
    TEXT:Ernst Reuter Platz 7, 10587 Berlin
    LATLON: 52.5127183,13.3217624
 - 9
    ID:201
    TEXT:Ernst Reuter Platz 9, 10587 Berlin
    LATLON: 52.5129359,13.3203243
```

Figure 1: Example of a document indexed in the geocoding system.

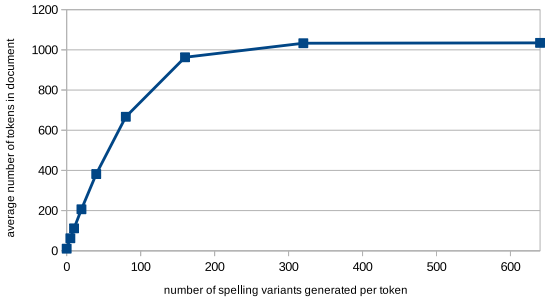


Figure 2: Average number of tokens per document for various amounts of spelling variants.

natim has been set up with OpenStreetMap data for Europe as the baseline geocoding system. Internally Nominatim uses a PostGIS (PostGIS, 2017) enabled PostgreSQL (PostgreSQL, 2017) database. After the preprocessing, this database contains assembled addresses along with their parent-child relationships: A house number level address is the child of a street level address, which in turn is the child of a district level address, etc. This database is used to extract address documents that are indexed in Elasticsearch, as defined in (Clemens, 2015a) and (Clemens, 2015b). Note that in this paper, the geocoding of only house number level addresses is evaluated. Therefore, though OpenStreetMap data also contains points of interests with house number level addresses, only their addresses but not their names have been indexed. Similarly, no parent level address elements, such as streets, postal code areas, cities, or districts have been indexed into Elasticsearch. All house number addresses with the same parent have been consolidated into one single document. Every house number have thereby been used as a key to specify the respective house number level address. Figure 1 shows an example document containing two house numbers 7 and 9, along with their WGS84 latitude and longitude coordinates and spelled-out addresses. The TEXT field of the document is the only one indexed; the TEXT fields mapped by the house numbers are only used to assemble a human-readable result.

Because Elasticsearch retrieves full documents, and because the indexed documents contain multiple house number addresses, a thin layer around Elasticsearch is needed to make sure only results with house numbers specified in queries are returned. That is a non-trivial task, as given a query, it is not known upfront which of the tokens is specifying the house number. Therefore, this layer has been implemented as follows: First, the query is split into tokens. Next, one token is assumed to be the house number; a query for documents is executed containing all the other tokens. This is repeated for each token, trying out every token as a house number. Because each time only one token is picked to specify the house number, this approach fails to support house numbers that are specified in multiple tokens. Nevertheless, it is good enough for the vast majority of cases. For every result document returned by Elasticsearch the house number map is checked. If the token assumed to be the house number happens to be a key in that map, the value of that map is considered a match and the house number address is added to the result set. Finally, the result set is returned. As edit distances are specified in the query to Elasticsearch, this layer allows enabling edit distances easily: A parameter passed to the layer is forwarded to Elasticsearch, which then also returns documents with fuzzily matching tokens. Also note, that as house numbers are used as keys in the documents, neither edit distances nor spelling variants are supported on house numbers. That, however, is a natural limitation: If a query specifies a different house number than the one intended, especially if it is a house number that exists in the data, there is no way for a geocoding system to still match to the right house number value.

Having the baseline systems Nominatim and Elasticsearch supporting edit distances set up, the next step is to create a similar system that indexes spelling variants. For that, the spelling variants to be indexed need to be defined first. HERE Technologies, the company behind the HERE geocoding system (HERE, 2017), provided logs of real user queries issued against the various consumer offerings of the company, like their website or the applications for Symbian, Windows Phone, Android and iOS mobile phones. The log contained data from a whole year and included queries users have issued along with results users chose to click on. For this paper, a user click is considered the selection criterion of a result, linking input queries to their intent, i.e. the addresses users were querying for. Given such query and result pairs, first both were tokenized and the Levenshtein distance (Levenshtein, 1966) from every query token to every result token was computed. With edit distances at hand, the Hungarian method (Kuhn, 1955) was used to align every query token to a result token.

From these computations, several observations were extracted:

1. Some query tokens are superfluous as they do not match (close enough) to any result token. Such tokens are ignored.
2. As the result is a fully qualified address, result tokens have an address element type, such as city, street, house number, or country. Thus, for each query, the query format, i.e., which address elements were spelled out in what order, is known.
3. Some query tokens matched to result tokens are misspelled. Thus, for each spelling variant of a token, the specific spelling mistake made is known. For this paper, the following classes of spelling variants were considered:
   - *inserts:* Characters are appended after a tailing character, prepended before a leading character, or inserted between two characters, e.g., *s* is often inserted between the characters *s* and *e*, as apparently the double-s in *sse* sounds like a correct spelling for many users.
   - *deletes:* Characters are removed after a character that is left as the tailing one, before a character that is left as the leading one, or between two characters that are left next to each other, e.g., *oa* between the characters *r* and *d* are often deleted, as users often abbreviate *road* as *rd*.
   - *replacements:* One character is replaced by a different character, e.g., *ß* is often replaced by an *s* in user queries so that *Straße* becomes *Strase* instead.
   - *swaps:* Two consecutive characters are swapped with each other, e.g., *ie* is often times swapped into *ei*, as, to users, both sounds may seem similar.

Thus from each query and result pair, the query format used as well as the set of spelling variations can be deduced. Doing so for all queries while counting the occurrences of each query format and each spelling variation results in a statistical model capable of two things: For a given token the model can determine the possible spelling variations, each with their observed count or relative probability. Also, out of a set of available address elements, the model can select and order elements such that the resulting choices correspond to formats human users use, each with their observed count or relative probability too. Because the spelling mistakes made as well as the query formats used are Pareto distributed (Arnold, 2015), the model contained a long tail of mistakes and formats used only very few times. To reduce the noise, the model was cleansed by stripping off the 25% of all observations from the long tail of rare spelling mistakes and query formats. In addition to that, all query formats that did not contain a house number were removed too, as the goal was to generate queries for addresses with house numbers. Because the log used is, unfortunately, proprietary, neither the log nor the trained model can be released with this publication. However, having a similar log of queries from another source enables the creation of a similar model.

Having the user model at hand, the spelling variants for indexing were derived as follows: Given a document to be indexed, its *TEXT* field was tokenized first. Next, for each token *N* most common spelling variants were fetched from the model and appended to the field. Thus, the field contained both the properly spelled tokens as well as *N* spelling variants for each token. Every house number level address from Nominatim was extracted from the database, augmented with spelling variants and indexed in Elasticsearch. For *N* the values 5, 10, 20, 40, 80, 160, 320, and 640 were chosen. Note that given a model, especially for short tokens, the number of applicable spelling variations is limited. In most extreme cases for a given token no spelling variant can be derived from the model at all. Figure 2 shows the resulting token counts of the *TEXT* field for every *N*. There is only a minor increase between indexing 320 and 640 spelling variants, as with 320 spelling variants almost all observed variants have already been generated.

An interesting aspect of the described approach is that, besides lowercasing, no normalization mechanisms have been exploited. While users often choose to abbreviate common tokens like street types, or avoid choosing the proper diacritics, the idea is that the model would observe common replacements of *Avenue* with *Av.*, or *Straße* with *Strasse* and generate according spelling variants for indexing. Like with the index without spelling variants, house numbers are not modified in any way here.

In total, three geocoding systems were set up with exactly the same address data indexed: Nominatim as the reference geocoder for OpenStreetMap data, Elasticsearch with documents containing aggregated house numbers and a layer to support edit distances, and Elasticsearch with indexed spelling variants. While the edit distance was specified at query time by specifying a parameter to the layer wrapping Elasticsearch, for the various numbers of indexed spelling variants distinct Elasticsearch indices have been set up. As the same layer has been used for all Elasticsearch based indices, the setup supported the possibility to query an index with spelling variants indexed while allowing an edit distance at the same time, the-

reby evaluating the effect of the combination of the two approaches.

## 3 MEASURING THE PERFORMANCE

To evaluate the geocoding systems for precision - the ratio of responses not containing results not queried for, and recall - the ratio of responses containing only the right result, 50000 addresses have been sampled from the data used in these systems. Using the generative user model, for each address a query format has been chosen so that the distribution of the query formats corresponded to the observed distribution of the query formats preferred by users. Next, for each query one to five query tokens have been chosen to be replaced with a spelling variant. Again, spelling variants picked were distributed in the same way the spelling variants of human users were distributed. Thus common query formats, and frequent spelling mistakes were often present in the test set, while rare query formats and rare spelling variants were selected rarely. This way six query sets with 50000 queries each have been generated. One contained all tokens in their original form, while the others had between one and five query tokens replaced with a spelling variant respectively. Note that not always a query had the desired number of spelling variants: The token to be replaced with a spelling variant was chosen at random. For some tokens, as discussed, no spelling variant can be generated by the model. These tokens were left unchanged, making the query contain fewer spelling variants than anticipated. Also, sometimes the house number token was chosen to be replaced. Given the set up of the documents in the indices, where house numbers are used as keys in a map, such queries had no chance of being served properly. This, however, does not pollute measurement results, as it equally applies to all systems evaluated. Because generated queries and indexed addresses originate from the same Nominatim database, both share the same unique identifier. Therefore, inspecting the result set of a response for the result a query has been generated from is a simple task.

Each test set was issued against indices with 5, 10, 20, 40, 80, 160, 320, and 640 indexed spelling variants, against the index with no spelling variants that allowed edit distances of 1 and 2, and against the two baselines: An index with neither spelling variants indexed nor edit distances allowed, as well as Nominatim. Additionally, each query set was issued against the combination of the two approaches: Indices with spelling variants indexed were queried so that edit distances were allowed. For every query set, responses were categorized into three classes: (i) Responses that yielded no result, (ii) responses that yielded only the correct result the query was generated from, and (iii) responses containing at least one wrong result that - as the query was not generated from that - was not the query intent. As the classes cover all possible cases and do not overlap, it is sufficient to consider two of the three metrics: While the ratio of cases in (ii) exactly is the recall of a geocoding system, the ratio of responses with wrong results in (iii) allows computing precision with ease.

The fact is that knowing the distributions of spelling variants, it is possible to calculate how many responses will include the expected result without any measurement: The portion of spelling variants indexed is exactly the portion of spelling variants in queries that an index will be able to serve. There is, however, no simple way to calculate the precision, as it heavily depends on the data and how ambiguous queries with spelling variants become. This, in turn, makes it impossible to compute the recall as it is defined for this experiment. These measurements allow observing the development of both metrics while the number of indexed spelling variants or the number of supported edit distances are increased.

## 4 RESULTS

Figure 3 shows an overview of the recall and inversed precision of some select systems tested. The blue chart denotes the performance of Nominatim, while the green chart denotes the performance of Elasticsearch with neither spelling variants indexed nor edit distances allowed. On the left-hand side, for recall, Nominatim performs slightly better for queries with no or one spelling mistake. That is most likely due to the normalization mechanisms that are built into Nominatim, but missing in Elasticsearch: Likely, a chunk of commonly made spelling variants can be handled through normalization. For no spelling mistake, both charts show higher recall compared to the red and yellow charts plotting the recall of the index with 320 spelling variants per token indexed, and the recall of enabling the edit distance of one, respectively. These two systems gain a slightly lower recall, due to their slightly lower precision visible on the right-hand side. As discussed, more queries become ambiguous when spelling variants are indexed, or edit distances allowed, leading to more responses containing results that the respective query was not generated from. As expected, the more spelling variants there are present in queries, the more recall

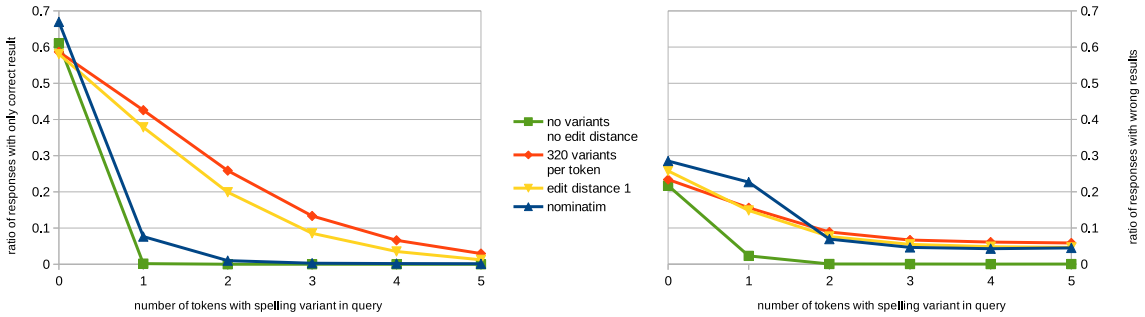


Figure 3: Recall (left, more is better) and inversed precision (right, less is better) of select systems.

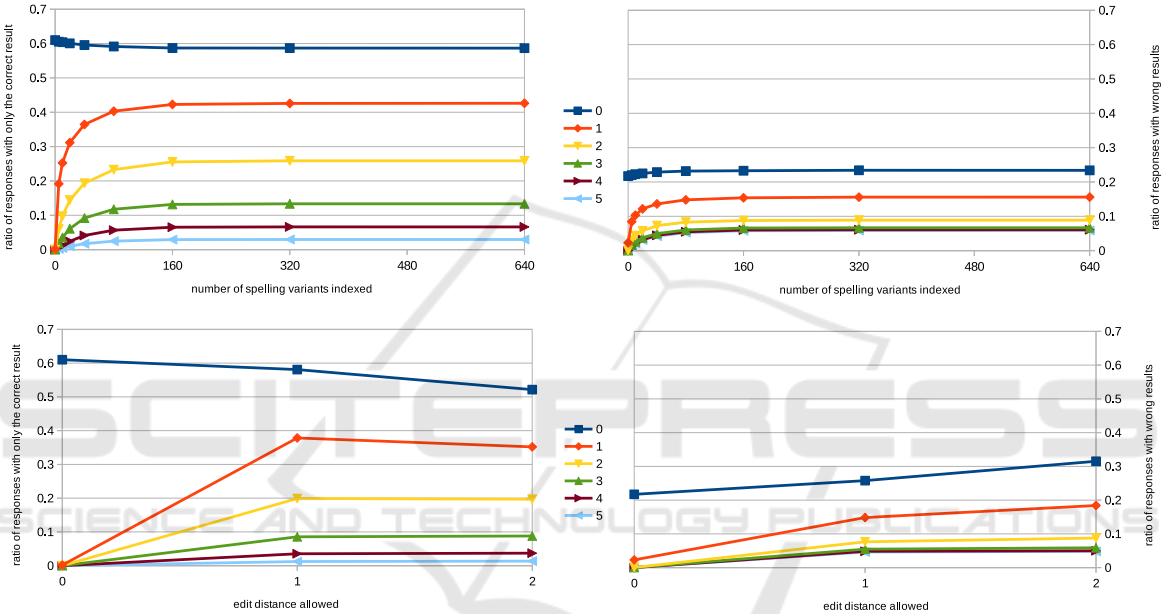


Figure 4: Detailed overview on the performance of indexing spelling variants and allowing edit distance.

drops. Without exception, the index with 320 spelling variants per token indexed outperforms the index allowing an edit distance of one. For zero or one spelling variant Nominatim has the lowest precision, returning most of the responses with results the query did not query for, while, as expected, the most strict system with neither spelling variants index nor edit distances allowed performs the best. The other two systems - one allowing an edit distance of one, the other indexing 320 spelling variants for each token - perform very similarly. Thereby, for no spelling variants the system allowing an edit distance of one performs slightly worse, while for any number of spelling variants in the queries, it performs slightly better. However, the margin of difference between the two systems with regards to precision is minor, compared to the margin of difference for the same two systems for recall. Generally, both the ratio of replies with the correct result as well as the ratio of replies containing wrong results drop more, the more spelling variants are present in the query. That is due to the number of replies with no result growing, as neither system can process queries containing too many spelling mistakes.

The detailed experiment results are denoted in Figure 4. Each line in the charts represents the development of recall or inversed precision on a specific test set. The legend specifies the allowed number of spelling errors in the queries of a test set. The top two charts show the recall and the inversed precision of the six test sets depending on how many spelling variants per token were indexed. Unsurprisingly, the more spelling errors a query contains, the less responses with only correct results are retrieved. At the same time, however, the ratios of responses containing wrong results decrease. Thus, the more errors a

user makes, the less results are discovered by the system overall. This behavior is also observable on the bottom two charts showing the performance on the six test sets depending on what edit distance was allowed. Interestingly, increasing the allowed edit distance to be greater than one does not improve the recall on any test set. At the same point, it worsens the precision, as with an allowed edit distance of two more candidates fit to the queries, resulting in more responses containing wrong results. That symptom is not observable when indexing spelling variants. As discussed, indexing 640 spelling variants for every token of the document almost maxed out the total number of tokens generated. The observation is that for every test set indexing more spelling variants leads to a clear improvement of recall. This pattern is also observable when enabling an edit distance of one, though to a lesser extent. Overall, on every test set, both the recall of the index containing spelling variants is greater compared to the index allowing edit distances, while their precision is of similar size. The blue chart showing the test set containing zero spelling variants visualizes the impact of allowing edit distances or indexing spelling variants on the left-hand side best: Indexing spelling variants or allowing an edit distance both reduce the recall by a similar degree, though the recall of the geocoder indexing 640 spelling variants is slightly greater compared to enabling an edit distance of one.

Table 1: Configurations yielding best recall.

| variants in query | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| variants indexed | 0 | 640 | 320 | 160 | 320 | 640 |
| edit distance | 0 | 0 | 0 | 1 | 1 | 1 |
| only correct result | 61% | 43% | 26% | 13% | 6% | 3% |
| also wrong result | 21% | 16% | 9% | 7% | 7% | 6% |

In Table 1 the combinations of indexed spelling variants and allowed edit distances that led to best results with regards to recall for the various test sets are listed. Interestingly, the number of spelling variants in the index varies between 160 and 640. That is an artifact of the random generation of queries. The numbers also show that for one or two spelling errors in queries, allowing edit distances on top of indexed spelling variants does not lead to any improvement of recall. Only if three or more query tokens are misspelled, a combination of indexed spelling variants and edit distance are yielding a better performance.

## 5 CONCLUSION

As already observed in previous papers, here too, Nominatim does not handle spelling mistakes well. Using a statistical model to derive and index common spelling variants, however, has proven to be a viable approach to serve queries with spelling errors.

Compared to allowing edit distances, it yields more responses containing only the right result, while only marginally increasing the number of responses with wrong results. Interestingly, this approach implicitly incorporates any standardization logic that would be of help: Exactly those abbreviations or misspelled diacritics are indexed as spelling variants that are commonly made. The experiment also suggests to index all possible spelling variants a cleansed model can generate: No number of indexed spelling variants smaller than that turned out to be the optimum beyond which performance of the index would degrade. Also, while indexed spelling variants outperform edit distances on all query sets, a combination of the two showed slightly better results for queries with many typos.

Going forward, it is worth investigating how spelling variants can be indexed without obtaining a statistical user model first. In this paper user clicks were used to learn how often and which typos are made. Users, however, can only click on results they receive. Thus, a query token may be spelled so significantly different, that the system will not present the proper result to the user. Even if that spelling variant would be common, without a result to click on, no model could learn that spelling variant so that it can be indexed. Further, the set of supported spelling variants might be defined more precisely. The model could learn more circumstances of an edit, like, e.g., four or more characters that surround an observed edit, as opposed to two characters only. Pursuing this idea to its full extent, a model could learn specific spelling variants for specific tokens instead of edits that can be applied in different scenarios, though doing so would probably require to utilize normalization mechanisms independent of the model. Another interesting study would be to measure how much such a model degrades over time. Assuming that user behavior changes, it is likely that the kind of spelling errors common at one point in time will no longer be common some time later. Thus, if a geocoder only relies on indexed spelling variants, its performance would be reduced over time.

## REFERENCES

Arnold, B. C. (2015). *Pareto distribution*. Wiley Online Library.

Borkar, V., Deshmukh, K., and Sarawagi, S. (2000). Automatically extracting structure from free text addresses. *IEEE Data Engineering Bulletin*, 23(4):27–32.

Can, L., Qian, Z., Xiaofeng, M., and Wenyin, L. (2005). Postal address detection from web documents. In *International Workshop on Challenges in Web Information Retrieval and Integration, 2005. (WIRI'05)*, pages 40–45. IEEE.

Clemens, K. (2013). Automated processing of postal addresses. In *GEOProcessing 2013: The Fifth International Conference on Advanced Geograhic Information Systems, Applications, and Services*, pages 155–160.

Clemens, K. (2015a). Geocoding with openstreetmap data. *GEOProcessing 2015: The Seventh International Conference on Advanced Geograhic Information Systems, Applications, and Services*, page 10.

Clemens, K. (2015b). Qualitative Comparison of Geocoding Systems using OpenStreetMap Data. *International Journal on Advances in Software*, 8(3 & 4):377.

Clemens, K. (2016). Comparative evaluation of alternative addressing schemes. *GEOProcessing 2016: The Eighth International Conference on Advanced Geograhic Information Systems, Applications, and Services*, page 118.

Coetzee, S., Cooper, A., Lind, M., Wells, M., Yurman, S., Wells, E., Griffiths, N., and Nicholson, M. (2008). Towards an international address standard. 10th International Conference for Spatial Data Infrastructure.

Davis, C. and Fonseca, F. (2007). Assessing the certainty of locations produced by an address geocoding system. *Geoinformatica*, 11(1):103–129.

Drummond, W. (1995). Address matching: Gis technology for mapping human activity patterns. *Journal of the American Planning Association*, 61(2):240–251.

Duncan, D. T., Castro, M. C., Blossom, J. C., Bennett, G. G., and Gortmaker, S. L. (2011). Evaluation of the positional difference between two common geocoding methods. *Geospatial Health*, 5(2):265–273.

Elastic (2017). Elasticsearch. https://www.elastic.co/products/elasticsearch.

Fang, L., Yu, Z., and Zhao, X. (2010). The design of a unified addressing schema and the matching mode of china. In *Geoscience and Remote Sensing Symposium (IGARSS), 2010*. IEEE.

Fitzke, J. and Atkinson, R. (2006). Ogc best practices document: Gazetteer service-application profile of the web feature service implementation specification-0.9. 3. *Open Geospatial Consortium*.

Ge, X. et al. (2005). Address geocoding.

geo poet (2017). http://geo-poet.appspot.com/.

Goldberg, D., Wilson, J., and Knoblock, C. (2007). From Text to Geographic Coordinates: The Current State of Geocoding. *URISA Journal*, 19(1):33–46.

Google (2017). Geocoding API. https://developers.google.com/maps/documentation/geocoding/.

HERE (2017). Geocoder API Developer's Guide. https://developer.here.com/rest-apis/documentation/geocoder/.

Kuhn, H. W. (1955). The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97.

Levenshtein, V. I. (1966). Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10, pages 707–710.

Mayrhofer, A. and Spanring, C. (2010). A uniform resource identifier for geographic locations ('geo'uri). Technical report, RFC 5870, June.

National Imagery and Mapping Agency (2004). Department of Defense, World Geodetic System 1984, Its Definition and Relationships with Local Geodetic Systems. In *Technical Report 8350.2 Third Edition*.

OpenStreetMap Foundation (2017a). Nomatim. http://nominatim.openstreetmap.org.

OpenStreetMap Foundation (2017b). OpenStreetMap. http://wiki.openstreetmap.org.

PostGIS (2017). http://postgis.net/.

PostgreSQL (2017). http://www.postgresql.org/.

Robertson, S., Zaragoza, H., and Taylor, M. (2004). Simple BM25 extension to multiple weighted fields. In *Proceedings of the thirteenth ACM international conference on Information and knowledge management*, pages 42–49. ACM.

Roongpiboonsopit, D. and Karimi, H. A. (2010). Comparative evaluation and analysis of online geocoding services. *International Journal of Geographical Information Science*, 24(7):1081–1100.

Salton, G. and Yang, C.-S. (1973). On the specification of term values in automatic indexing. *Journal of documentation*, 29(4):351–372.

Salton, G., Yang, C.-S., and Yu, C. T. (1975). A theory of term importance in automatic text analysis. *Journal of the American society for Information Science*, 26(1):33–44.

Sengar, V., Joshi, T., Joy, J., Prakash, S., and Toyama, K. (2007). Robust location search from text queries. In *Proceedings of the 15th annual ACM international symposium on Advances in geographic information systems*, page 24. ACM.

Srihari, S. (1993). Recognition of handwritten and machine-printed text for postal address interpretation. *Pattern recognition letters*, 14(4):291–302.

Universal Postal Union (2017). http://www.upu.int.

what3words (2017). what3words. https://map.what3words.com/.

Yahoo! (2017). BOSS Geo Services. https://developer.yahoo.com/boss/geo/.

Yandex (2017). Yandex.Maps API Geocoder. https://tech.yandex.com/maps/geocoder/.

Yang, D.-H., Bilaver, L. M., Hayes, O., and Goerge, R. (2004). Improving geocoding practices: evaluation of geocoding tools. *Journal of medical systems*, 28(4):361–370.