Master Thesis

Geoparsing Dutch Archival Descriptions. *A Pipeline for Extracting and Linking Toponyms to the Linked Open Data Cloud.*

H.J. Schouten | 5717094

Applied Data Science, MSc

Utrecht University

July 11, 2021

First Supervisor

Prof. dr. A.H.L.M. (Toine) Pieters

Second Supervisor

Ivar Troost MSc

Table of Contents

1 Introduction

1.1 Introduction

History and historical events are defined by both time and space. Historians who try to reconstruct this past need historical sources originating from the same time and space as the history they study. To find relevant sources, the digitalizing of archival indexes and records and the current large-scale digitalizing of historical sources made this much easier than it was before. However, while it is easy to find sources through time, as dates are usually part of the metadata of archival records, finding sources in space is much more challenging. This is also the case in The Utrecht Archives (Het Utrechts Archief, HUA), a regional archive for the province of Utrecht in The Netherlands. In this thesis project, I will try to extract geographic named entities (toponyms) and link them to the Linked Open Data (LOD) cloud by linking to Wikidata, where the coordinates can be found. The goal is to develop a pipeline to perform this process called geocoding or geoparsing for all current 21 million archival descriptions in HUA. This project takes place within the context of the Utrecht Time Machine (UTM).[1] The UTM is a collaboration between several archival (including HUA), governmental, and research institutions in the city of Utrecht, with the goal to virtually bring the history of Utrecht back to life and to make archives and historical sources better accessible for both historians as for other people interested in the archives. As part of these efforts, archives and historical data sets are currently being made accessible as Linked Open Data (LOD). As the UTM is part of the European Time Machine project,[2] under whose umbrella in various cities similar Time Machine projects are currently running, it might become possible to connect and query all participating European archives together. Adding a geographical dimension to HUA will contribute to better accessible archives

[1] https://utrechttimemachine.nl/

[2] https://www.timemachine.eu/

and, therefore, contribute to the UTM, and hopefully in the future to the European Time Machine project.

1.2 Relevance of geoparsing archival descriptions

In archives where the location is not explicitly defined in the metadata in a standardized form, such as for the most significant part of Het Utrechts Archief (HUA), finding sources about, or related to, a specific geopolitical or other geospatial location is difficult. Finding an archival record for a specific location is currently only possible by a textual query on the archival descriptions. This search method is problematic for at least four reasons: (1) it is unknown for researchers whether the geographic location they refer to with a certain toponym is the same as in the relevant archival description, like names and spelling of the names of streets, buildings, and cities, change over time. Even (2) if the researchers know the exact toponym during the period they are searching for, they do not know whether or not this historical toponym is used in the archival descriptions. Besides this, (3) place names are, especially in the past, not uncommon as person surnames; this increases the noise in textual geographic search results. These three problems are getting worse, (4) if researchers are looking for archival records originating from multiple locations within a particular area, e.g. a neighbourhood or arbitrary spatial area; researchers should take into account the problems mentioned above, for all toponyms (such as streets) within the area of interest. Therefore, being able to query coordinates and standardized locations would significantly improve the accessibility of archives and help historians and other interested people find what they are looking for. As historians attach great importance to authenticity, it is essential that while standardizing the locations, the original reference will be untouched. Standardizing locations and respecting the authenticity can be accomplished by linking the toponyms in the archival descriptions to a knowledge base such as Wikidata, where the coordinates, among other information, are stored, including historical variants and dates about the respective location. Because items in Wikidata have an arbitrary URI (Uniform Resource Identifier), which consist of the letter ‘Q’ with a number (e.g. Q803 for the municipality of Utrecht), this method is persistent for the future name or spelling changes of the city or street, as the URI is not dependent on the toponym at the moment of creation. The knowledge base to which will be linked can be more local as well and could at the same time also serve as a hub to other more general knowledge bases. An example of this is Adamlink[3], which is part of the Amsterdam Time Machine.

1.3 Proposal for new pipeline

As part of the UTM, a previous pilot project at The Utrecht Archives, HUA Op de kaart (On the map)[4], made a start in geoparsing archival descriptions by geocoding the digitalised HUA image collection (Beeldbank). The results were visualised on a map, making it easy for people to browse through the collection and find historical photographs or images of specific parts of the city. A gazetteer (which is in the context of geocoding, a list of toponyms) containing contemporary street names in Utrecht was used to look up mentions of streets, together with regular expressions that matched common affixes of street names as straat and laan. Regular expressions were intended to extract mentions of streets outside the geographical extent of the gazetteer. Because of this approach, many references could be missed because of spelling variations or incomplete lists, or streets might be extracted only partially, which made it difficult or impossible to link to the knowledge base.

In this project, I will try to overcome the limitations of the previous project and improve its results by taking a different approach. I will use Named Entity Recognition (NER) to extract toponyms from the archival descriptions and a custom rule-based Entity Linking algorithm that uses a sentence encoder for fuzzy matching found toponyms to toponyms in a local database. The NER model and the EL algorithm together form a geoparsing pipeline for geoparsing Dutch archival descriptions. Initially, the research question focused only on how NER and EL could be employed to geoparsing archival descriptions. During the process, the question altered to the following, which also takes the performance into account, and the absence of gold standard training data:

How, and to what extent can we extract toponyms Dutch archival descriptions in Het Utrechts Archief, with a fine-grained fine-tuned Named Entity Recognition (NER) model, and link them to the Linked Open Data (LOD) cloud, without gold standard training data?

The hypotheses are that (1) it is possible to extract the geographic named entities in the archival descriptions with a (fine-grained) fine-tuned NER model, and (2) link them to the LOD-

[3] Adamlink.nl

[4] http://hualab.nl/opdekaart/index.php

cloud where the location coordinates are present, and (3) that this can be done without having gold standard training data.

The outline of this thesis is as follows: Before I will dive into the pipeline, in the next chapter, I will sketch the background of this method by describing related work and discussing the NER and Entity Linking task in general; then, in the data chapter, I will explain the structure of (Dutch) archives and the contents of archival descriptions; next, in the methods chapter, I will explain and describe the two parts of the pipeline I developed, as well as the process of the creation of these two parts. This chapter will then be followed by an evaluation chapter, after which in the discussion and conclusion chapter, I will point out shortcomings and directions for further development of the current pipeline.

2 Background and Related Work

2.1 Related work

Besides the earlier discussed HUA Op de kaart project, it appears that there no similar research has been done on applying NER and Entity Linking to this particular corpus type: (Dutch) archival descriptions. However, the techniques to do this are widely applied to various corpora. Because archival descriptions are a mixture of modern texts with historical concepts and references, I will briefly describe examples of NER and geoparsing applied to both modern and historical, both Dutch and non-Dutch (mostly English) corpora.

In modern Dutch texts, Schraagen, Brinkhuis and Bex[5] used an out-of-the-box NER tool called Frog, on Dutch criminal complaints to evaluate whether this could be used for automatically processing filed complaints regarding cases of online fraud. Manual evaluation of 227 entities learned that the tool performed poorly on the specific corpus, with an overall F1-score of .38. Childish, Tudhope and Wansleeben[6] applied NER to Dutch archaeological reports to extract mentions of artefacts, archaeological features, materials, places and time entities. As they

[5] Schraagen, Marijn, Matthieu Brinkhuis, and Floris Bex. 2017. “Evaluation of Named Entity Recognition in Dutch Online Criminal Complaints.” *Computational Linguistics in the Netherlands Journal* 7: 3–16.

[6] Vlachidis, Andreas, Douglas Tudhope, and Milco Wansleeben. 2021. “Knowledge-Based Named Entity Recognition of Archaeological Concepts in Dutch.” In *Metadata and Semantic Research*, 1355:53–64. Nature Publishing Group. https://doi.org/10.1007/978-3-030-71903-6_6.

did not have gold standard training data, the researchers created a rule-based classifier based on a domain-specific conceptual model, thesauri and glossaries, in combination with handcrafted rules based on word features such as word case, morphological features and grammar. With their approach, they reached an F1-score of .67. The archaeological reports corpus shares some characteristics with archival descriptions, as these texts contain both current and historical names and concepts; a rule-based approach for this corpus in the absence of training data, however, did not yield very well results. In both Dutch examples, however, no explicit geoparsing was done. To geocode Spanish contemporary news articles, Molina-Villegas et. al.[7] used a training set for training a NER model of 1233 news articles in which all 5870 locations were annotated. Additionally, a corpus of 165,354 articles was used for training word embedding, which they used for entity disambiguation (linking a location reference to a location in the real world). They queried the Overpass API from OpenStreetMap and retrieved all points of interest (e.g. streets, monuments and shops) in a 200 metres radius, which were together converted to one document vector through the embeddings, which they called pseudo documents. As the original news article (original document) was also converted to a vector, they could use the euclidean and cosine distance to compare the pseudo documents to the original document and choose the location that belonged to the nearest pseudo document. Their NER model reached an F1-Score of 0.6995. In 74.62% of the time, the nearest pseudo document was the correct real-world location. Gupta and Nishu took a different approach for a similar task concerning English news articles [8]. The authors used a NER model from SpaCy to annotate the locations in 10,000 news article sentences from a local newspaper. To combine locations to form a more specific location annotation, such as combining a street, a number and a city to form an address, they used hard-coded rules which used Part-of-Speech (POS) annotations (created with a SpaCy POS model). With this training data, a BERT model was fine-tuned to annotate the locations, resulting in an F1-score of .77. Next, they used the OpenStreetMap API and the Google Places API to geocode the extracted

[7] Molina-Villegas, Alejandro, Victor Muñiz-Sanchez, Jean Arreola-Trapala, and Filomeno Alcántara. 2021. “Geographic Named Entity Recognition and Disambiguation in Mexican News Using Word Embeddings.” *Expert Systems With Applications* 176: 114855. https://doi.org/10.1016/j.eswa.2021.114855.

[8] Gupta, Sarang, and Kumari Nishu. 2020. “Mapping Local News Coverage: Precise Location Extraction in Textual News Content Using Fine-Tuned BERT Based Language Model.” In Proceedings Ofthe Fourth Workshop on Natural Language Processing and Computational Social Science, 155–62. https://doi.org/10.18653/v1/2020.nlpcss-1.17.

locations; to reduce costs, the OpenStreetMap was used for finding geopolitical entities (which were defined by the SpaCy NER model), and Google Places for geocoding only for looking up the other locations. The performance of the geoparsing part is unknown as the authors did not include an evaluation. On historical corpora, NER and Entity Linking are applied as well.[9] In the Netherlands, Peris et al. extracted city names from historical newspapers.[10] They used multiple out-of-the-box NER tools in a voting classifier, as well as a gazetteer with city names of interest, with which they queried the articles. Van Veen, Lonij and Faber[11] used NER on Dutch historical newspaper articles and aimed to link the NE-s to the LOD-cloud (Wikidata and DBpedia). In this project, after they tried a rule-based approach, they used a Support Vector Machine (SVM) classifier to link entities to a database containing a DBpedia dump. The SVM took beside the actual entity also other metadata into account, such as year and place of publication. While they reached an accuracy of 83% correct links, they estimated that it should be possible to reach a 95% accuracy in an automated process.

2.2 Named Entity Recognition

As becomes clear above, NER algorithms can consist of simply gazetteers and regular expressions or rules that are hard-coded or learned by machine learning. Besides these algorithms, there are also NER models that are deep learning based. [12] NER algorithms and models usually classify the named entities in at least three categories: Persons, Locations, Organizations.[13] For named entities that do not fall into one of these three categories, there is

[9] Won, Miguel, Patricia Murrieta-Flores, and Bruno Martins. 2018. “Ensemble Named Entity Recognition (NER): Evaluating NER Tools in the Identification of Place Names in Historical Corpora.” *Frontiers in Digital Humanities* 5 (March). https://doi.org/10.3389/fdigh.2018.00002.

[10] Peris, Antoine, Willem Jan Faber, Evert Meijers, and Maarten Van Ham. 2020. “One Century of Information Diffusion in the Netherlands Derived from a Massive Digital Archive of Historical Newspapers: The DIGGER Dataset.” *CyberGeo 2020* (January). https://doi.org/10.4000/cybergeo.33747.

[11] Van Veen, Theo, Juliette Lonij, and Willem Jan Faber. 2016. “Linking Named Entities in Dutch Historical Newspapers.” In *Communications in Computer and Information Science*, 672:205–10. Springer Verlag. https://doi.org/10.1007/978-3-319-49157-8_18.

[12] Yadav, Vikas, and Steven Bethard. 2019. “A Survey on Recent Advances in Named Entity Recognition from Deep Learning Models.” http://2016.bionlp-st.org/tasks/bb2.

[13] Goyal, Archana, Vishal Gupta, and Manish Kumar. 2018. “Recent Named Entity Recognition and Classification Techniques: A Systematic Review.” *Computer Science Review* 29: 21–43. https://doi.org/10.1016/j.cosrev.2018.06.001.

often a miscellaneous category. NER models trained or build for a specific task or domain can include other or different categories, as is the case with the NER model I trained for this project.

2.3 Entity Linking

Linking the found entities to real-world locations, however, is a more challenging task. A knowledge-base or gazetteer is essential in this process because geographic locations (coordinates) are practically impossible to generalize. This means that toponyms, as they are exacted from a (historical) text, need to be matched and linked to contemporary and standardized (spelled) names and locations. Until recently, this is done with rule-based algorithms, usually with hard coded rules, string distance metrics, a combination of both, or with a machine learning algorithm with various distance metrics as input data.[14] [15] A recent development in this field of toponym resolution is the application of deep learning methods.[16] As with general NLP tasks such as NER, there is even a Python library implementing this neural network model for toponym matching, which comes with a pre-trained model that can also be fine-tuned.[17] These neural networks are trained to compare one reference to a candidate location and decide if the two strings are similar. Since it is costly to try each entity with all locations on a gazetteer in such a model, a candidate selection step is needed only to compare relevant locations. While these models seem attractive, they have some downsides. I will name three of those shortcomings: First is the need for specific training data to train Entity Linking models. This is indeed also true for NER algorithms; however, training data for this task can be relatively quickly (semi-)automatically created, which is not the case for EL. As with the creation of NER training data, it is, of course, possible to look for known named entities from a list. However, since some

[14] Recchia, Gabriel, and Max Louwerse. 2013. "A Comparison of String Similarity Measures for Toponym Matching." In *COMP 2013 - ACM SIGSPATIAL International Workshop on Computational Models of Place*, 54–61. https://doi.org/10.1145/2534848.2534850.

[15] Santos, Rui, Patricia Murrieta-Flores, and Bruno Martins. 2018. "Learning to Combine Multiple String Similarity Metrics for Effective Toponym Matching." *International Journal of Digital Earth Output*. Vol. 00.

[16] Wang, Jimin, and Yingjie Hu. 2019. "Are We There yet? Evaluating State-of-the-Art Neural Network Based Geoparsers Using EUPEG as a Benchmarking Platform." In* Proceedings of the 3rd ACM SIGSPATIAL International Workshop on Geospatial Humanities, GeoHumanities 2019. Association for Computing Machinery, Inc.* https://doi.org/10.1145/3356991.3365470.

[17] Hosseini, Kasra, Federico Nanni, and Mariona Coll Ardanuy. 2020. "DeezyMatch: A Flexible Deep Learning Approach to Fuzzy String Matching." In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, 62–69. https://doi.org/10.18653/v1/2020.emnlp-demos.9.

toponyms, e.g. street names such as ‘Kerkstraat’ (Church Street), are very common, it is hard to link those occurrences to the correct entity, as multiple instances of Kerkstraat will be on the list. Second, not all models consider other entities, appearing near an entity, and thus missing valuable information such as the city where a street might be located. Third, entity linking models, such as the Entity Linker from SpaCy, are often trained on Wikidata labels and descriptions, which makes the linker dependent on, first the actual presence of a description, and second the completeness of this description, i.e. that it contains enough contextual information about the entity to be linked correctly, such as the city in which an entity occurs. If a description is missing or incomplete, which I noticed is not uncommon, the model will have trouble linking these entities.

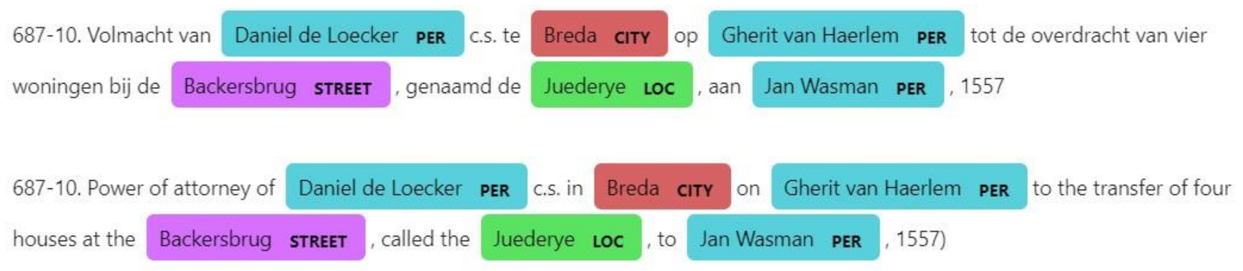
In this thesis, I will use a linking algorithm that will use fine-grained NER tags to filter candidate entities, after which the toponym will be linked to the closest candidate label (or alternative label). I will elaborate more on the specifics of the algorithm in the methods chapter.

3 Data (On Archival descriptions)

The collection of The Utrecht Archives contains currently around 21 million archival descriptions, covering a multitude of historical documents and other sources. The variety of records is wide; documents vary in nature, place and time of creation, somewhere between 1000 years ago and today. These documents are all indexed and described with a short text that informs about the content of a particular document or folder of documents that form one archival record group, such as letters of a particular person. The archives are structured in a hierarchical way in which sources from one institution or person, or sources of the same type or topic, are grouped in the same collection. Usually, multiple groupings are used in one archival entry, e.g. all notarial deeds are grouped, in which these deeds are further grouped by creators, the notaries, in which deeds are again grouped by the type or the time of creation.[18] Besides the individual records, grouped records have a group description as well. On average, descriptions have 11 words (s=16.8 words), but their length varies much per record (group) and record type. Sometimes a concise description is sufficient, e.g. for birth records, while more complex records require a

[18] ***34-4 Notarissen in de stad Utrecht 1560-1905***, Het Utrechts Archief, https://hetutrechtsarchief.nl/collectie/609C5BA4A1024642E0534701000A17FD. Accessed on 10-6-2021.

more detailed description, e.g. for specific medieval charters. For grouped records, the descriptions of the individual records are usually shorter since the longer group description already gave more general information about the group's contents. If documents are, for example, grouped by place of creation or spatial extent that the documents cover, usually only the group description contains information about the particular place. Concerning the language of descriptions, in general, collections of documents are usually written in modern Dutch, explaining what is in the collection or group of records, while descriptions covering only one document such as a charter, beside a few modern Dutch words from the archivist, most of the description consists of verbatim quotes from the source. Quoted elements can include the (historical) toponyms, but sometimes the modern toponym is also given. Figure 1 shows an example archival description is shown, with the toponyms and persons tagged. While Haerlem is the name of a city, it should not be regarded as such in this context, as it is part of a person name. This example shows the downside of a naive gazetteer approach, as this name would probably be erroneously regarded as a toponym in such an approach. Therefore a more advanced method that takes the context into account is needed.


687-10. Volmacht van Daniel de Loecker PER c.s. te Breda CITY op Gherit van Haerlem PER tot de overdracht van vier woningen bij de Backersbrug STREET , genaamd de Juederye LOC , aan Jan Wasman PER , 1557

687-10. Power of attorney of Daniel de Loecker PER c.s. in Breda CITY on Gherit van Haerlem PER to the transfer of four houses at the Backersbrug STREET , called the Juederye LOC , to Jan Wasman PER , 1557)

Figure 1. Example description (original in Dutch) with toponyms tagged CITY, STREET and LOC (location), and PER (persons). 'Haerlem' is not a toponym in this context.

4 Methods: The Pipeline

4.1 General Overview

As mentioned before, the goal is to create a pipeline for geoparsing archival descriptions from HUA. The pipeline consists of two parts: a Named Entity Recognition part and an Entity Linking part. The creation and development of these two parts will be described separately, in their respective order in the pipeline. Because during this project, the methods changed, and this is part of the research project, I will describe this process in the sections below. I can, however, make some general notes about the followed method in advance. For the NER model, I first weighted two possible models for fine-tuning, after which I created training data from archival descriptions and categorised entities into various location categories with the help of a pre-trained NER model and a gazetteer. I then fine-tuned the NER model with this training data. Because the results were below the targeted performance, I improved the training data with a different pre-trained NER model after the results were more satisfactory. For the Entity Linking component, I first developed an algorithm for narrowing down the number of candidate matches in a database using the NER-tags and fuzzy matching techniques. However, this first algorithm turned out to be sometimes too slow and sometimes yielded strange results. After 2.7 seconds, 'Weerafdeling', erroneously classified as a street in Utrecht, was linked to a street called 'D' (Wikidata Q18986121), in a small town in the province of Groningen. Therefore I made a second version of this algorithm in which I used a sentence encoder to speed up and improve the search operation. Because there is no gold standard validation set, I will try to indicate the performance of the pipeline, as well as the individual components; I will script-wise validate on a large subset of the annotated Beeldbank dataset and manual inspect a small random subset of the archival descriptions in the NER test set.

4.2 NER model

4.2.1 NER models

For this project, I looked at two different out-of-the-box NER models. The first is SpaCy, which is a complete NLP pipeline with pre-trained models in various languages, with models for all major NLP tasks.[19] *T*he (pre)processing pipeline can be entirely altered by the user: (custom) components or models can be added, removed or temporarily disabled, and all models can be fine-tuned or retrained for which an API is included. The second model is FLAIR, which is a transformer-based neural network.[20] These types of models are state-of-the-art for NLP tasks. On the Dutch Conll-2002 dataset, it reached an F1-score of .9225. However, the downside is that

[19] https://spacy.io/

[20] Alan Akbik, Tanja Bergmann, Duncan Blythe, Kashif Rasul, Stefan Schweter, and Roland Vollgraf. 2019. "FLAIR: An Easy-to-Use Framework for State-of-the-Art NLP." In *Proceedings of NAACL-HLT 2019: Demonstrations*, 54–59. https://www.aclweb.org/anthology/N19-4010.pdf.

they are very large (many parameters) and therefore much slower in predicting named entities than other models. The processing of 70,000 archival descriptions took FLAIR about 45 minutes, while SpaCy needed only four to five minutes to do this. Because HUA has 21 million descriptions, the processing of the entire archive would take more than a week. Moreover, training a model like this is time-consuming as well, which could be problematic, as experiments with different hyper-parameters and datasets will take much time. Because of the flexibility of the SpaCy package and its speed, I chose to use the SpaCy model in this project. The goal for this NER model is to reach an F1-score above .9.

4.2.2 Training

According to the SpaCy documentation, adding new NER categories to their NER model needs a few thousand to a million training examples.[21] As a start, I downloaded an arbitrary number of 130,000 HUA descriptions to use as training, validation and test data. From this, after the labelling process, I wanted to have at least 100,000 valid examples I could use for training (70,000), validation (15,000) and testing (15,000), which turned out to be sufficient to train the new categories. Because it was impossible to annotate all these descriptions manually within the limited time window for this project, I took a more automated approach. This approach consisted of annotating with gazetteers for the fine-grained categories and annotating with existing NER models for other NER categories. Because this was an iterative process with two cycles, I will describe this training process and the updating of the training dataset as such.

[redacted]*First Cycle*

Training data

In the first cycle, I created the training data by looking up names from gazetteers for cities, villages and municipalities in the Netherlands, streets in the Province of Utrecht, provinces in the Netherlands and neighbourhoods in Utrecht. These lists came from Wikidata, but I also collected lists with historical municipalities (including historical spelling variants) from Histograph.*[22] From the Wikidata items, I also collected the aliases besides the labels to make sure that the most common alternative references were also tagged. The categories added in this dataset are

[21] SpaCy, "Chapter 4: Training a Neural Network Model." *Advanced NLP with SpaCy*. Accessed June 14, 2021. https://course.spacy.io/en/chapter4.

[22] https://github.com/histograph/histograph

STREET, NEIGH (neighbourhood), CITY, and PROV (province). Because I wanted to keep the original SpaCy NER labels (CARDINAL, DATE, EVENT, FAC, GPE, LANGUAGE, LAW, LOC, MONEY, NORP, ORDINAL, ORG, PERCENT, PERSON, PRODUCT, QUANTITY, TIME, WORK_OF_ART), I also let the original SpaCy Dutch model (large) process the descriptions. In instances where the SpaCy model double-tagged a token, or a range of tokens, such as a street labelled as location, I preferred the more fine-grained label. In cases where a token or a range of tokens classified as a custom tag but where those tokens were fully or partially tagged as a person as well, I preferred the custom tag, unless the person name was longer than the custom tag. By doing this, I could reduce the problem of person names that are (in part) place names, mistakenly tagged as, for example, a city, by the quite rigorous approach with the gazetteers. For this reason, I wanted to keep the person category in the model. Descriptions containing a toponym that is the name present in multiple categories, e.g., a street and a city, were removed from the training data. Doing so reduced the training set, but this was not problematic because of the large number of initial training examples. Moreover, removing these ambiguous instances would likely improve training.

Training

After training with this dataset for eight epochs when no improvement appeared, it turned out that the model did not get as good as I wanted it to be. Especially the original SpaCy tags were performing poorly. Especially the LOC and FAC, but also ORG and PERSON tags were insufficient (F1-score between .50-.70). However, the new tags appeared to be quite well (all with an F1-score between .80 and .90). Because the SpaCy model was able to learn the new tags quite good, I decided to stick with SpaCy but change the other tags' training data since I did not reach the performance I hoped to get.

[redacted]*Second Cycle*

Training data

For the second cycle, I used the FLAIR model instead of the original SpaCy model to add other tags besides my earlier mentioned location tags. Since the FLAIR model only classifies into the basic NER categories (person, location, organisation and miscellaneous), at the same time, the number of categories reduces, which could result in better performance since those categories might be better defined and less ambiguous. During an inspection of the newly added labels, I noticed that many place names were present in the LOC category that I thought should be in my original gazetteer. It turned out that here minor spelling differences were causing problems. Besides this, I noticed that other well-defined locations, such as countries and rivers, were detected by the model. As I started to manually assign the LOC entities into the right more specific category (if applicable), I took the opportunity to add at the same time new categories for RIVER, WATER (water body which is not a canal or river) and COUNTRY to the existing categories. For these categories, I extracted gazetteers from Wikidata and labelled the LOC-entities with these lists, after which I relabelled all remaining LOC entities that were found more than one time. Since the DATE category was not in the FLAIR label scheme, it was missing from the training data. However, because I thought this category could be helpful for the EL process, I created a SpaCy Matcher for finding dates. Such a Matcher allows patterns of words or word types (Part-Of-Speech or NER tags, e.g.) and regular expressions. I added two of those patterns for this task, which I put in the appendix.

Training

The model reached much higher F1-scores with this dataset after twelve epochs for the test set, as shown in Table 1. Almost all crucial categories score above or just below the .9, and some get very close to 1. The LOC category still performs poorly, which might be caused by the streets, cities, and other locations still in this category. Tokens classified by this model as LOC should thus be handled with caution. The PER(son) category performs below the performance target as well, perhaps partially caused by initials separated from the surname in some examples. With this fixed, the score will probably be higher.

	Precision	*Recall*	*F1-score*
DATE	0.98	0.98	0.98
CITY	0.97	0.98	0.97
PROV	0.94	1.00	0.97
STREET	0.94	0.92	0.93
RIVER	0.86	0.94	0.90
NEIGH	0.89	0.89	0.89
COUNTRY	0.85	0.87	0.86
PER	0.80	0.77	0.78
WATER	0.78	0.78	0.78
ORG	0.75	0.73	0.74
LOC	0.64	0.51	0.57
Overall	***0.93***	***0.92***	***0.93***

Table 1. Final NER model evaluation on test set (n=15,000), specified by type.

4.3 Entity Linking

4.3.1 Introduction

As pointed out above, there are a few disadvantages of trained entity linking models: the need for training data which is hard to obtain; the sometimes disregarding of the context of an entity; the limited number of entity features on which the models are trained (usually only the name or textual description from a knowledge base). Therefore, in combination with the limited amount of time for this thesis, which reduced the space to experiment, I decided to write an entity-lookup algorithm. Writing an algorithm myself was easier because of the fine-grained NER tags. With these tags, the linking algorithm narrows down the number of possible candidate entities from a local (partial) Wikidata database for an entity that needs to be linked. The database (DB) consists of (nearly) every entity located within the Netherlands. For each entity, the following attributes are included in the database: administrative entity in which the entity is located, the entity label, the entity description, all entity alternative labels, the street on which an entity is located (optional), the address of the entity (optional), the location (wdt:P276, optional), and the entity type(s). All labels are in Dutch.

4.3.2 Linking Algorithm

[redacted]*First Algorithm*

The algorithm processes one description at a time. The first step in narrowing down the number of candidate entities is by looking up the city where an entity is located (if the city is mentioned in the description). The algorithm does this by checking for an exact full or partial match in a city subset of the DB. If this has not been found, with the Levenshtein distance, the algorithm chooses the closest match if the match is above a certain threshold (maximum score is 100 for an exact match). The Levenshtein distance is a metric that calculates the minimum number of character edits to change one string into the other.[23] For these lookups, first, the algorithm queries the city subset, after which, if there is no result, it tries the entire DB.

It makes a subset of the DB when it finds a city, which now includes only the entities located within this city (or cities if applicable). Next, the algorithm links the other entities in the following order: STREET, LOC, NEIGH, RIVER, WATER. The actual linking process is the same as for cities, except that in the case of multiple candidates, the intersection of these entities is used to narrow the number of candidates. If for one street entity multiple candidates are found, these candidates are temporarily stored. These street candidates can help to identify the correct candidate entity for a different entity, such as a building if again multiple options were present. In this case, the algorithms select the building where the street on which the building is located and is one of the candidate street entities. Besides the building entities, also the street entity can be linked to a Wikidata item. If all entities are processed, and multiple candidates exist, these are stored and returned separately to be further processed. Entities that could not be linked to an entity in the DB are returned separately and need to be looked up on Wikidata directly. Currently, I did not implement this in the algorithm. However, the algorithm performance is disappointing, as are its results, which were sometimes strange, as the example given in the chapter introduction. The Levenshtein distance probably causes this poor performance because the distance is calculated each time one string is compared to a candidate. Therefore, this can be a computationally intensive task, especially for large DB subsets (large cities, e.g.) and longer strings. In combination with a chain of subsets that sometimes needed to be queried, the processing time for one description could increase from 0.2 seconds up to 2 or sometimes even 4 seconds, after which a proper match is not guaranteed.

[23] Recchia, Gabriel, and Max Louwerse. 2013. “A Comparison of String Similarity Measures for Toponym Matching.” In *COMP 2013 - ACM SIGSPATIAL International Workshop on Computational Models of Place*, 54–61. https://doi.org/10.1145/2534848.2534850.

[redacted]*Second Algorithm with Sentence Encoder*

To widen the bottleneck of the first version of the EL algorithm, I looked for a solution in which I could drastically cut back the DB search time. I decided to step away from the string (edit) distance metrics, as those metrics would run into the same scalability problems as the Levenshtein distance. An alternative is to encode strings into an n-dimensional space from a (sub-)word or character embedding, after which the distance can be quickly calculated between two or more strings. Because encodings are only numbers describing the document or string, the encoding is independent of the string to which it is compared and thus needs to be calculated once for the whole DB in advance. Only the string which is to be compared with the DB needs to be encoded before a query. Therefore I chose this method to speed up the algorithm. Because of time limitations, I could not extensively compare different encoders. After a quick and manual evaluation of a few candidates, I choose a pre-trained multi-lingual 768-dimensional transformer sentence-encoder.[24] The Python *sentence_transformers* library in which the model is loaded has a built-in search function. This function takes a list of encoded queries and a matrix with encoded candidates (up to a million instances) as inputs and returns the desired number of top results with a similarity score for each result. Using this method, it took only 870 ms to compare one query to 300,000 candidates, while this took 17 seconds when using the Levenshtein distance.

In essence, this second algorithm stays the same as the first, as, for a description, I first match the city and then the other entities in a subset of the DB with only items of respective NER type located in that city. However, there are some differences: first, I now only have one lookup method, which is the encoding distance described before, instead of the exact and partial string match, and Levenshtein fuzzy match. Second, as this method already gave very accurate results, the inference and intersection step for dealing with multiple equal results are currently not implemented because of limited time. In the case of multiple results, all the results are returned. When dealing with places or towns and municipalities, no choice is made in favour of municipalities as this might better be decided on afterwards, and because both entity types are sometimes used interchangeably or inconsistently, which could cause problems in the sub-setting part. Third, besides a threshold parameter that indicates whether the (top) result is considered a match or not (current default score .95), I added a near-hit threshold (default .7). Suppose there are no scores above the primary threshold, but items with a score above the near-hit threshold are

[24] https://huggingface.co/sadakmed/distiluse-base-multilingual-cased-v2

present. In that case, it is considered that: (1) possibly the correct Wikidata item is in the database, but an alternative name or label is closer to the entity (first only the main labels are checked, as in the first algorithm), or (2) the NER model might only have partially extracted the entity. The latter option is checked first by matching the entity itself with additional characters before and after the entity in the archival description (a parameter defines the number of characters, default is 30). The top results are joined with the earlier top results, after which the candidate with the best match to the extended entity is picked. If, however, no candidates remain after the join, the alternative labels, or, if this also leaves no results, subsequently, the descriptions are queried. This version is much more stable and faster per description in terms of performance, with .145 seconds (std. ± .078 seconds, seven runs of 10 descriptions).

5 Evaluation

Here I will evaluate both components of the pipeline separately, beginning with the NER model. As there is no real gold standard validation set for neither the NER model nor the EL algorithm, I will evaluate the components on a small sample of 600 manually validated archival descriptions and 15,000 descriptions of the linked Beeldbank dataset. The Beeldbank dataset can be seen as a benchmark test, as it shows how the pipeline and its components perform in relation to the approach of the HUA Op de kaart project. It is, however, not a gold standard validation set because it is not manually validated and did not include as many location types as are included in the new pipeline. To deal with the presence of many short archival descriptions in the archive, the evaluation subset is not entirely random. To prevent a subset with only non-informative short descriptions, I took a random sample of 200 long descriptions (>150 characters), 200 medium length descriptions (50-150 characters), and 200 short descriptions (<50 characters).

5.1 NER

5.1.1 Beeldbank NER

The results of the NER model are presented in Table 2. From the locations in the Beeldbank subset (n locations=18,359 in 15,000 descriptions), 85% (15,567) are retrieved by the NER model. While missing 15% of the locations, the model extracted 25,492 locations that were not present in the subset. However, many of those were cities or villages; these are implicitly present in the Beeldbank set, as they are included in the street locations. When these places were filtered out, 7,615 additional locations remained, 42% of the total number of original locations. From those new locations, 3,751 are streets. Further specifications of the extra locations types are in Table 3.

	n	**% of n Beeldbank locations**
Beeldbank Locations	18,359	-
Found BB locations	15,567	84.8
Not found BB locations	2,792	15.2
Extra locations (total)	25,492	138.8*
Extra locations (without cities)	7,615	41.5*

*Table 2. NER results. *Not including the original Beeldbank locations; the percentage is only taken from the number of original Beeldbank locations.*

Rank	**Wikidata ID**	**n**	**Label**
1	Q2039348	15915	municipality of the Netherlands
2	Q79007	3751	street
3	Q1852859	1113	populated place in the Netherlands
4	Q12280	638	bridge
5	Q486972	449	human settlement
6	Q200334	410	bell tower
7	Q532	325	village
8	Q16970	315	church building
9	Q16917	163	hospital
10	Q2795367	153	neighbourhood in Utrecht
11	Q1484611	137	buurtschap
12	Q515	128	city
13	Q123705	115	neighbourhood
14	Q523166	103	canal
15	Q19844914	100	university building
16	Q41176	89	building
17	Q3947	75	house
18	Q13539802	67	place with town rights and privileges
19	Q1060829	62	concert hall
20	Q174782	60	square

Table 3. Top 20 new Beeldbank location types.

5.1.2 Archival Descriptions NER

The manual annotated small archival descriptions subset (n=600), the performance can be expressed in precision, recall and F1-scores. However, the standard metrics are strict; both the extracted entity and the tag must be exactly the same as the gold standard. This stringent evaluation method is helpful during training but is not very informative when evaluating a NER model. Therefore I evaluate the model with the evaluation rules used in the SemEval-2013 Task 9.1[25], as implemented by the Nervaluate Python library[26]. Entities are scored on four categories: strict, complete match of string and tag; exact, exact string match regardless of tag; partial, partial string match regardless of tag; type, correct tag for a string, regardless of whether or not the string has an exact match. These scores per NER type and the overall scores for all types together are presented in Table 4. A complete table including the raw counts on which the scores are based can be found in appendix A. For better understanding and comparison of the results, the F1-scores per tag per score category are displayed in Figure 2. In general, the model can extract most entities while also assigning the correct type (F1 strict=.82). For type matches (part or full string match with type correct), the F1-score is .85. These scores are pretty good, as the linking algorithm can handle these incomplete string matches (currently only if the incomplete match is not an entity in the database itself). The model extracts most locations and other entities at least partially, as well (F1 partial=.87, exact=.85). The top three best-performing tags are, considering strict and type matches, CITY (F1 strict=.93, type=.93), STREET (F1 strict=.90, type=.90) and PER(son) (F1 strict=.88, type=.90). If we widen the scope to partial matches to see whether or not the locations of these tags are extracted at all (however not tagged correctly), the best tags are WATER (F1 exact=1., but n=3), CITY (F1 exact=.94), NEIGH (F1 partial=.94) and STREET (F1 partial=.90).

The DATE tag performs much worse than in training; the F1 score dropped with 0.14 from .98 at the test set after training to .84 (type and strict). This difference might be caused by the inability of the regular expressions used for creating this category to capture all dates. However, th e performance of the LOC category is surprisingly good, as it had a test F1 of 0.57. In this set,

[25] Isabel Segura-Bedmar, Paloma Martínez, and María Herrero-Zazo. 2013. “SemEval-2013 Task 9: Extraction of Drug-Drug Interactions from Biomedical Texts (DDIExtraction 2013).” In *SEM 2013 - 2nd Joint Conference on Lexical and Computational Semantics*, 2:341–50. http://www.cs.york.ac.uk/semeval-2013/task9/.

[26] https://github.com/MantisAI/nervaluate

it scores similar on strict matches (F1 = .6) but has an exact F1 score of .72 and a partial F1 score of .81. The lower type and strict score are expected, as the LOC tag is a noisy category, as noted in the previous chapter.

In general, it needs to be noted that in some descriptions, especially the long and sometimes medium-length descriptions, the location city could be mentioned more than once. In some instances, only one of those toponyms was correctly extracted and tagged. As only one of those mentions is enough to link the description, the scores cannot necessarily be interpreted as the linkability of the descriptions. One other fascinating insight from the manual inspection, which can not be found in the numbers, is that the model seemed not only able to extract entities from Dutch descriptions but from descriptions written in Latin, German and English as well. However, this feature is not a complete surprise, as descriptions written in these languages are (probably) present in the training data. Additionally, the German and English descriptions, in particular, were often book descriptions. These descriptions are in a standardized format, which the model might have (partially) learned.

		Precision	**Recall**	**F1-score**	**N locations to be extracted of type**	**N locations extracted of type**
Overall	***type***	***0.88***	***0.83***	***0.85***	***1790***	***1691***
	partial	***0.90***	***0.85***	***0.87***		
	strict	***0.84***	***0.80***	***0.82***		
	exact	***0.87***	***0.82***	***0.85***		
CITY	type	0.94	0.92	0.93	264	259
	exact	0.95	0.93	0.94		
	partial	0.96	0.95	0.95		
	strict	0.92	0.90	0.91		
COUNTRY	type	0.73	0.67	0.70	12	11
	exact	0.91	0.83	0.87		
	partial	0.95	0.88	0.91		
	strict	0.64	0.58	0.61		
DATE	type	0.91	0.81	0.86	773	693
	exact	0.89	0.80	0.84		
	partial	0.90	0.81	0.85		
	strict	0.89	0.80	0.84		
LOC	type	0.62	0.61	0.62	80	79
	exact	0.72	0.71	0.72		
	partial	0.82	0.81	0.81		
	strict	0.61	0.60	0.60		
NEIGH	type	0.56	0.56	0.56	16	16
	exact	0.94	0.94	0.94		
	partial	0.97	0.97	0.97		
	strict	0.56	0.56	0.56		
ORG	type	0.83	0.74	0.78	234	209
	exact	0.77	0.69	0.73		
	partial	0.86	0.76	0.81		
	strict	0.71	0.64	0.67		
PER	type	0.89	0.91	0.90	298	307
	exact	0.88	0.90	0.89		
	partial	0.90	0.92	0.91		
	strict	0.87	0.89	0.88		
PROV*	type	0.50	0.50	0.50	8	8
	exact	0.75	0.75	0.75		
	partial	0.88	0.88	0.88		
	strict	0.50	0.50	0.50		

RIVER*	type	0.75	0.67	0.71	9	8
	exact	0.88	0.78	0.82		
	partial	0.94	0.83	0.88		
	strict	0.75	0.67	0.71		
STREET	type	0.88	0.92	0.90	93	98
	exact	0.88	0.92	0.90		
	partial	0.90	0.95	0.92		
	strict	0.85	0.89	0.87		
WATER*	type	0.67	0.67	0.67	3	3
	exact	1.00	1.00	1.00		
	partial	1.00	1.00	1.00		
	strict	0.67	0.67	0.67		

*Table 4. Performance of the NER model on the archival descriptions subset (n descriptions=600), scored according to SemEval-2013 metrics (see text). The total number of locations present per type in the evaluation set, and the number of locations actually extracted per type are shown in the last two columns. *NOTE: less than ten entities of this tag are present in the evaluation set.*

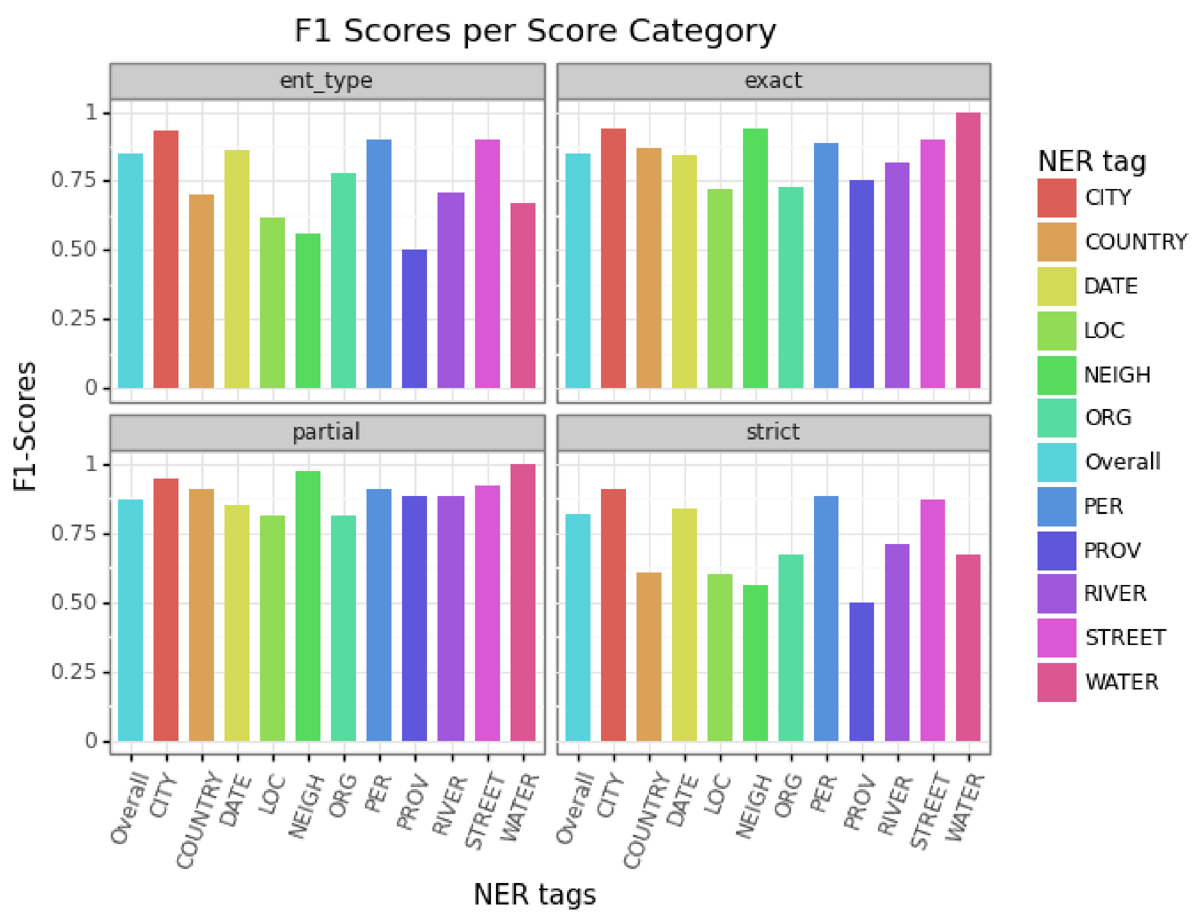


Figure 2 . NER model F1-scores per NER tag, per score category on the archival description subset. Note that for PROV, RIVER and WATER, less than 10 entities were present in the evaluation set.

5.2 Entity Linking

5.2.1 Beeldbank EL

In Table 5, the results of the Entity Linking algorithm on the Beeldbank set can be seen. The EL algorithm performs almost perfectly, with only 0.5% (73) of the Beeldbank locations, found by the NER model, not (correctly) linked. From the extra locations, only 1% (258) is not linked. This means that the current pipeline would equally link 84.7% of the linked locations in the Beeldbank dataset. However, it should be noted that we do not know to what extent the locations in the Beeldbank set are correct.

	n	**%**
Beeldbank Locations	18,359	-
Found BB locations		***% of found BB location***
Found BB locations	15,567	100
Found and linked BB locations	15,544 (84.7% of all BB locations)	99.5
Found and not linked BB locations	73	0.5
Found Extra locations		***% of found Extra locations***
Extra locations (total)	25,492	100
Extra locations (total) linked	25,234	99.0
Extra locations (total) not linked	258	1.0
Found Extra locations (no cities)		***% of found Extra locations (no cities)***
Extra locations (without cities)	7,615	100

Table 5. Entity Linking results on Beeldbank set. Reported extra locations without cities are probably the true extra locations, as the Beeldbank dataset does not explicitly references cities; only streets are linked, and as streets are located in a city or municipality, these are implicitly linked.

5.2.2 Archival Descriptions EL

Unfortunately, the limited time frame prevented a comprehensive manual inspection of the links resulting from entities extracted by the NER model. I did, however, looked through the links. While I cannot show any scores, I can make some general notes about the quality of the links. First, I noticed that the algorithm is not performing as well on this set compared to the Beeldbank set. This seemed to be caused by the incomplete descriptions, i.e. if a description is part of a larger archival group or collection of documents, information about the city from which the archive originated is described somewhere higher in the hierarchy. Thus, the archivist did not write the location of a document in every description, resulting in difficulties for the linking algorithm when only using the description. Second incorrect links were caused by the limited geographical scope of the database. I did not include locations outside the Netherlands in the database, while these were present in the descriptions. In some instances, the algorithm returned an incorrect location from the database instead of a non-match. Third, problems arose for incorrect assigned tags or incomplete extracted entities that appear in the dataset. In one instance, from ‘Departement van den Landbouw’ (Department of Agriculture), ‘Departement van den’ was tagged as ORG, while ‘Landbouw’ was tagged as STREET. Furthermore, in the database, a street named Landbouw was found and was thus linked as such. Finally, I noticed that for locations in the Netherlands, which was the majority of the locations, the links were almost always correct in case an entity was linked. Especially when a city was mentioned, the results were outstanding. If the municipality was abolished, there were some difficulties, as no proper subset could be made for that city. On Wikidata, all entities which once fell within this former administrative entity are located in the new municipality.

6 Discussion and conclusion

In this thesis, I developed a Dutch archival description geoparsing pipeline consisting of a fine-grained NER model and an EL algorithm, which is, to my knowledge, the first of its kind. The research question was how and to what extent we can extract toponyms Dutch archival descriptions in Het Utrechts Archief, with a fine-grained fine-tuned Named Entity Recognition (NER) model, and link them to the Linked Open Data (LOD) cloud, without gold standard training data. With the hypotheses that (1) it is possible to extract the geographic named entities in the archival descriptions with a (fine-grained) fine-tuned NER model, and (2) link them to the LOD-cloud where the location coordinates are present, and (3) that this can be done without having gold standard training data.

I trained the fine-tuned NER model on semi-automatically created training data. I created the training data by annotating archival descriptions with locations from a gazetteer and added other tags (location, person, organisation) to the descriptions with first SpaCy’s NER model, and later a FLAIR NER model, as the results with the former model were disappointing. Common unspecified locations annotated by the FLAIR model were split into different other categories, as cities and streets were among those locations. The final NER model can exact entities and categorise them into the following categories DATE, CITY, PROV, STREET, RIVER, NEIGH, COUNTRY, PER, WATER, ORG, LOC. The model scored a general F1-score of .82 for strict matches on a small archival description evaluation set, and the model was able to extract 85% of the Beeldbank locations, which is lower than the target score of .9. However, this score is much better than the scores of the geoparsing projects mentioned in the related work section, where the highest score was .77 from Gupta and Nishu, who also trained their custom model. They trained their model on 10,000 partially automatically annotated sentences, and Molina-Villegas et al. used only 1233 annotated articles, resulting in a NER F1-score of .67.

Manual inspection of the archival description dataset annotated by my NER model, showed the limitations of the automatically created training data, as some locations were incorrectly tagged while the context indicated the correct tag. In other instances, the locations not correctly capitalised were missed. Automatically created training data was needed, as many examples were necessary to train the new entity tags. However, now that the tags are trained, the next step could be to create a smaller manually corrected training set of 1000 or 2000 descriptions and fine-tune the current model. Because these descriptions can be first tagged by the current model and then corrected manually, this is not time-consuming. I think this could significantly improve the scores of the model to reach at least an F1-score of .9. If this performance, however, is not reached, training a different model, such as a FLAIR transformer model, could be considered. I can thus conclude that it is possible to extract geographic named entities from archival descriptions with a (fine-grained) fine-tuned NER model. As the model is trained without gold standard training data, I can also conclude that this is possible with my approach. I think the better performance compared to Gupta and Nishu, who also created their training data automatically, is at least in part caused by the much more extensive training dataset I used for training my NER model. However, I am convinced that my model can still improve by fine-tuning it on a smaller manually curated training set. Besides this, the initial training data creation can be improved by using the best existing model available, and, to improve the quality of the fine-grained categories, especially the general location category, matching those entities with a gazetteer should be accompanied with fuzzy matching, such as applied in the EL algorithm. This step would probably reduce the noise in the location category and improve the other categories as well.

The Entity Linking algorithm uses the fine-grained NER tags to link the found entities to locations in a local database, currently containing all locations in the Netherlands, extracted from Wikidata. In the final algorithm, a subset of candidate matches is created by filtering on the entity type (the NER tag), and if present in the extracted locations and linked, the city. To link a found entity to a location in the subset, the algorithm uses a sentence encoder. This encoder encodes all locations in the database (done before running the pipeline) and the entity which is to be linked. The encodings can then be used to calculate the distance between the entity and the candidates. As described earlier, several steps are in place to handle ambiguous results. This linking method is similar to Molina-Villegas et al., however, I use a pre-trained embedding model and first filter the candidate locations based on location type and, if possible, a city mentioned in the text. Besides this, I encode the label, alternative label or description of a candidate entity and use this for finding the correct match, instead of encoding a range of entities nearby a candidate location, as is done by Molina-Villegas et al. This algorithm can correctly link 99.5% of the Beeldbank locations extracted by the NER model. This performance is much better than the permanence of Van Veen, Lonij and Faber in linking entities from Dutch historical newspapers who used an SVM model for determining links and reached 85% accuracy, but they linked persons and organisations as well. This score also surpasses Molina-Villegas et al., who accurately linked 74.62% of the extracted locations; however, as they linked locations within Mexico, their geographical extent was much larger than in this thesis, which possibly makes linking more difficult. However, the performance of my model on the archival description set was, after visual inspection, not as well as on the Beeldbank. In the Beeldbank dataset, every description contains the city, making it easier to create subsets, which results in better links, as not many locations with the same name are present in the same city. In regular archival descriptions, this is usually not the case. These descriptions are often part of a hierarchical organised record group or collection of archival pieces, in which information about a geographical extent of a group or collection, e.g. a city, is given in the description of such a group. This geographical information is usually not repeated in every description below. Therefore, the algorithm should be extended with a feature to take into account the locations mentioned in the higher descriptions. Possibly it could use the city, or cities, named in these descriptions as a default city for creating subsets. For efficiency, archival descriptions might then be processed per archival group or collection in their hierarchical order. Moreover, the geographical extent of the current database was not sufficient, as locations in Germany and Belgium were extracted from descriptions and incorrectly or not linked. Therefore, the database should be extended with these countries, or the algorithm should contain a Wikidata query function to look up the location directly on Wikidata. The last main point of improvement of the EL algorithm is speed. Currently, the algorithm needs 0.145 seconds to match the entities of a description, which is quite long, especially as the actual distance calculation takes much less time. Creating subsets and looping through results currently makes up most of the processing time. This time could be reduced by streamlining these processes, using faster sub-setting methods and possibly merging or replacing operations. The loops and thresholds could, for example, be replaced by a machine learning model, trained on data created by the current linker, which selects the best match from a list of scores or encodings of the best candidates, possibly combined with additional features. If the mentioned improvements would be implemented, especially the propagation of locations mentioned in group descriptions, this approach is suitable for this problem. If the current algorithm is provided with the city context, the linking is near perfect, if of course, the correct candidate is present in the database. This algorithm then performs better than earlier mentioned solutions, which I think can be attributed to the combination with the fine-grained NER model, which significantly helps to narrow down the possible candidates.

In conclusion, this unique pipeline, with a fine-grained NER model, trained on partially automatically created training data, and the Entity Linking algorithm on top of this, are suitable for geoparsing Dutch archival descriptions. Furthermore, while this pipeline was initially created with The Utrecht Archives in mind, I think its usage extends to other Dutch archives as well, as the NER model learned to recognise locations from its context. Additionally, the results of the NER component includes other entity types as well, such as dates and organisations, and persons. Therefore, the pipeline might be extended to link these entities as well. This pipeline, while having room for improvement, can add a geographic dimension to The Utrecht Archives to a large extend and is at the same time a promising start for integrating the archive into the Linked Open Data cloud even further, contributing to the Utrecht and European Time Machine project.

7 References

Alan Akbik, Tanja Bergmann, Duncan Blythe, Kashif Rasul, Stefan Schweter, and Roland Vollgraf. 2019. "FLAIR: An Easy-to-Use Framework for State-of-the-Art NLP." In *Proceedings of NAACL-HLT 2019: Demonstrations*, 54–59. https://www.aclweb.org/anthology/N19-4010.pdf.

Ardanuy, Mariona Coll, Daniel C.S. Wilson, Katherine McDonough, Kasra Hosseini, Amrey Krause, and Daniel Van Strien. 2019. "Resolving Places, Past and Present: Toponym Resolution in Historical British Newspapers Using Multiple Resources." In *ACM International Conference Proceeding Series*, 37:6. Association for Computing Machinery. https://doi.org/10.1145/3371140.3371143.

Goyal, Archana, Vishal Gupta, and Manish Kumar. 2018. "Recent Named Entity Recognition and Classification Techniques: A Systematic Review." *Computer Science Review* 29: 21–43. https://doi.org/10.1016/j.cosrev.2018.06.001.

Gupta, Sarang, and Kumari Nishu. 2020. "Mapping Local News Coverage: Precise Location Extraction in Textual News Content Using Fine-Tuned BERT Based Language Model." In *Proceedings Ofthe Fourth Workshop on Natural Language Processing and Computational Social Science*, 155–62. https://doi.org/10.18653/v1/2020.nlpcss-1.17.

Hosseini, Kasra, Federico Nanni, and Mariona Coll Ardanuy. 2020. "DeezyMatch: A Flexible Deep Learning Approach to Fuzzy String Matching." In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, 62–69. https://doi.org/10.18653/v1/2020.emnlp-demos.9.

Isabel Segura-Bedmar, Paloma Martínez, and María Herrero-Zazo. 2013. "SemEval-2013 Task 9: Extraction of Drug-Drug Interactions from Biomedical Texts (DDIExtraction 2013)." In *SEM 2013 - 2nd Joint Conference on Lexical and Computational Semantics*, 2:341–50. http://www.cs.york.ac.uk/semeval-2013/task9/

Molina-Villegas, Alejandro, Victor Muñiz-Sanchez, Jean Arreola-Trapala, and Filomeno Alcántara. 2021. "Geographic Named Entity Recognition and Disambiguation in Mexican News Using Word Embeddings." *Expert Systems With Applications* 176: 114855. https://doi.org/10.1016/j.eswa.2021.114855.

Peris, Antoine, Willem Jan Faber, Evert Meijers, and Maarten Van Ham. 2020. "One Century of Information Diffusion in the Netherlands Derived from a Massive Digital Archive of Historical Newspapers: The DIGGER Dataset." *CyberGeo 2020* (January). https://doi.org/10.4000/cybergeo.33747.

Recchia, Gabriel, and Max Louwerse. 2013. "A Comparison of String Similarity Measures for Toponym Matching." In *COMP 2013 - ACM SIGSPATIAL International Workshop on Computational Models of Place*, 54–61. https://doi.org/10.1145/2534848.2534850.

Santos, Rui, Patricia Murrieta-Flores, and Bruno Martins. 2018. "Learning to Combine Multiple String Similarity Metrics for Effective Toponym Matching." *International Journal of Digital Earth Output*. Vol. 00.

Schraagen, Marijn, Matthieu Brinkhuis, and Floris Bex. 2017. "Evaluation of Named Entity Recognition in Dutch Online Criminal Complaints." *Computational Linguistics in the Netherlands Journal* 7: 3–16.

SpaCy, "Chapter 4: Training a Neural Network Model." *Advanced NLP with SpaCy*. Accessed June 14, 2021. https://course.spacy.io/en/chapter4

Veen, Theo van, Juliette Lonij, and Willem Jan Faber. 2016. "Linking Named Entities in Dutch Historical Newspapers." In *Communications in Computer and Information Science*, 672:205–10. Springer Verlag. https://doi.org/10.1007/978-3-319-49157-8_18.

Vlachidis, Andreas, Douglas Tudhope, and Milco Wansleeben. 2021. "Knowledge-Based Named Entity Recognition of Archaeological Concepts in Dutch." In *Metadata and Semantic Research*, 1355:53–64. Nature Publishing Group. https://doi.org/10.1007/978-3-030-71903-6_6.

Wang, Jimin, and Yingjie Hu. 2019. "Are We There yet? Evaluating State-of-the-Art Neural Network Based Geoparsers Using EUPEG as a Benchmarking Platform." In* Proceedings of the 3rd ACM SIGSPATIAL International Workshop on Geospatial Humanities, GeoHumanities 2019. Association for Computing Machinery, Inc.* https://doi.org/10.1145/3356991.3365470.

Won, Miguel, Patricia Murrieta-Flores, and Bruno Martins. 2018. "Ensemble Named Entity Recognition (NER): Evaluating NER Tools in the Identification of Place Names in Historical Corpora." *Frontiers in Digital Humanities* 5 (March). https://doi.org/10.3389/fdigh.2018.00002.

Yadav, Vikas, and Steven Bethard. 2019. "A Survey on Recent Advances in Named Entity Recognition from Deep Learning Models." http://2016.bionlp-st.org/tasks/bb2.

.

Appendix

Appendix A. Date Matcher patterns

Pattern 1

```
[{'TEXT': {'REGEX': '\d\d?'},
           'OP': '?'},
           {'LOWER': {'IN': ['januari',
                             'februari',
                             'maart',
                             'april',
                             'mei',
                             'juni',
                             'juli',
                             'augustus',
                             'september',
                             'oktober',
                             'november',
                             'december']}},
         {'TEXT': {'REGEX': "(\d{2,4}|'\d\d)"}}]
```

Pattern 2

```
[{'TEXT': {'REGEX': '\d\d?-\d\d?-\d{2,4}'}}]
```

Matcher

```
matcher = spacy.matcher.Matcher(nlp.vocab)
matcher.add('DATE', [Pattern 1, Pattern 2], greedy='Longest')
```

Appendix B. Extended Table NER Evaluation on Archival Descriptions

		correct	**incorrect**	**partial**	**missed**	**spurious**	**N locations to be extracted of type**	**N locations actually extracted of type**	**precision**	**recall**	**f1**
Overall	***type***	***1481***	***88***	***0***	***221***	***122***	***1790***	***1691***	***0.88***	***0.83***	***0.85***
	partial	***1475***	***0***	***94***	***221***	***122***	***1790***	***1691***	***0.90***	***0.85***	***0.87***
	strict	***1426***	***143***	***0***	***221***	***122***	***1790***	***1691***	***0.84***	***0.80***	***0.82***
	exact	***1475***	***94***	***0***	***221***	***122***	***1790***	***1691***	***0.87***	***0.82***	***0.85***
CITY	type	243	10	0	11	6	264	259	0.94	0.92	0.93
	exact	246	7	0	11	6	264	259	0.95	0.93	0.94
	partial	246	0	7	11	6	264	259	0.96	0.95	0.95
	strict	237	16	0	11	6	264	259	0.92	0.90	0.91
COUNTRY	type	8	3	0	1	0	12	11	0.73	0.67	0.70
	exact	10	1	0	1	0	12	11	0.91	0.83	0.87
	partial	10	0	1	1	0	12	11	0.95	0.88	0.91
	strict	7	4	0	1	0	12	11	0.64	0.58	0.61
DATE	type	629	1	0	143	63	773	693	0.91	0.81	0.86
	exact	615	15	0	143	63	773	693	0.89	0.80	0.84
	partial	615	0	15	143	63	773	693	0.90	0.81	0.85
	strict	615	15	0	143	63	773	693	0.89	0.80	0.84
LOC	type	49	23	0	8	7	80	79	0.62	0.61	0.62
	exact	57	15	0	8	7	80	79	0.72	0.71	0.72
	partial	57	0	15	8	7	80	79	0.82	0.81	0.81
	strict	48	24	0	8	7	80	79	0.61	0.60	0.60
NEIGH	type	9	7	0	0	0	16	16	0.56	0.56	0.56
	exact	15	1	0	0	0	16	16	0.94	0.94	0.94
	partial	15	0	1	0	0	16	16	0.97	0.97	0.97
	strict	9	7	0	0	0	16	16	0.56	0.56	0.56
ORG	type	173	24	0	37	12	234	209	0.83	0.74	0.78
	exact	161	36	0	37	12	234	209	0.77	0.69	0.73
	partial	161	0	36	37	12	234	209	0.86	0.76	0.81
	strict	149	48	0	37	12	234	209	0.71	0.64	0.67
PER	type	272	9	0	17	26	298	307	0.89	0.91	0.90
	exact	269	12	0	17	26	298	307	0.88	0.90	0.89
	partial	269	0	12	17	26	298	307	0.90	0.92	0.91
	strict	266	15	0	17	26	298	307	0.87	0.89	0.88
PROV	type	4	4	0	0	0	8	8	0.50	0.50	0.50
	exact	6	2	0	0	0	8	8	0.75	0.75	0.75
	partial	6	0	2	0	0	8	8	0.88	0.88	0.88
	strict	4	4	0	0	0	8	8	0.50	0.50	0.50
RIVER	type	6	2	0	1	0	9	8	0.75	0.67	0.71
	exact	7	1	0	1	0	9	8	0.88	0.78	0.82
	partial	7	0	1	1	0	9	8	0.94	0.83	0.88
	strict	6	2	0	1	0	9	8	0.75	0.67	0.71
STREET	type	86	4	0	3	8	93	98	0.88	0.92	0.90

	exact	86	4	0	3	8	93	98	0.88	0.92	0.90
	partial	86	0	4	3	8	93	98	0.90	0.95	0.92
	strict	83	7	0	3	8	93	98	0.85	0.89	0.87
WATER	type	2	1	0	0	0	3	3	0.67	0.67	0.67
	exact	3	0	0	0	0	3	3	10	10	10
	partial	3	0	0	0	0	3	3	10	10	10
	strict	2	1	0	0	0	3	3	0.67	0.67	0.67