university of groningen / faculty of arts

Multi-label classification of named entities

using Wikipedia's main topic classifications

J.F.P. (Richard) Scholtens

Master Thesis
Information Science
J.F.P. (Richard) Scholtens
s2956586
August 5, 2020

ABSTRACT

This research set out to see if it is possible to create an efficient and accurate fine-grained named entity multi-label classification system using Wikipedia's Main Topic Classifications as labels. Creating such a system will help other academics to improve existing data sets by adding new generalized labels for unknown named entities. By using four different data collections strategies information about 7991 entities spread out over 34 Main Topic Classifications for the Dutch language is gathered. Every one of these classifications has subcategories that hold information about a specific topic within the Main Topic Classification. Two annotators created a gold standard using MISC entities out of the SoNaR1-corpus, resulting in Kappa-score of 0.8182. Using fine-tuned bidirectional LSTM models in combination with GloVe and BERTje embeddings, one can perform multi-label classification based upon chunks of texts as input. The best text classification was an LSTM model using BERTje embeddings and resulted in an accuracy of 44,01%, a macro F1-score of 86,31 and a micro F1-score of 97,91%. Another LSTM model used GloVe embeddings resulting in an accuracy of 37,74%, a macro F1-score of 89,87%, and a micro F1-score 97,60%. All models performed the same multi-label named entity classification task and were evaluated on the gold standard resulting in an accuracy score of 70%, macro F1-score of 9,26%, and a micro F1-score of 12,45% for the LSTM using GloVe embeddings and accuracy of 91%, macro F1-score of 8,72% and a micro F1-score of 12,45% for the LSTM using BERTje embeddings. One must conclude that the Dutch Main Topic Classification articles are suitable to create a dataset for training a fine-grained multi-label named entity classification model if one takes the proper removing, merging and pruning steps. Therefore the performed methods succeeded in developing fine-grained entity multi-label classifications system but could not be considered robust enough. Therefor more information retrieval methods and optimization steps should be considered in future research. All documentation and code can be found on https://github.com/richardscholtens/Master_Thesis.

CONTENTS

1 INTRODUCTION

The named entity recognition and classification field lay its focus on recognizing and classifying named entities by recognizing specific properties. According to Cerri, Barros, and de Carvalho (2014), a typical classification system first step consists of creating a model that trains to find relationships for each training example by identifying features and linking them to a specific label. In the second step, the model takes in unclassified examples as input and tries to predict the correct label based upon its features. The difficulty of creating accurate classification systems lies in the number of prediction labels a model has to choose. By increasing the number of labels, the probability of choosing the correct label decreases and the chances of the labels overlapping increases. For instance, the study by Luyckx, Vaassen, Peersman, and Daelemans (2012) tried to detect suicide notes by labelling sentences with the correct emotion label. Instead of choosing the labels positive, negative, and neutral, they used fifteen different emotion labels so the severity of potential suiciders can be detected early on. For instance, someone who is angry and depressed can be more suicidal than someone who is only angry or only depressed. By choosing more fine-grained labels, medical personal can choose which people need are helped first. Choosing more multiple labels is called multi-label classification and originated from the investigation of the text categorization problem, where each document may belong to several predefined topics simultaneously according to Xia, Zhang, Yang, Li, Du, Wu, Fan, Ma, and Yu (2019).

Creating models that can predict fine-grained multi-classification with high accuracy seems to be challenging to develop as most traditional systems can only "identify a small set of types such as a person, location, organization or miscellaneous" according to Liu, Wang, Zhou, Wang, and Lee (2018). Developing higher accuracy named entity recognition and classification algorithms can help improve many different fields of expertise. However, the difficulty level also prevented to develop datasets that hold many generalized labels. For instance, according to Oostdijk, Reynaert, Hoste, and Schuurman (2013), the SoNaR-1 corpus holds one million Dutch sentences obtained out of conventional media but also webpages, chatboxes, and e-mail. These sentences recognize named entities and classify each recognized named entity to one of the six generalized classification labels, namely, persons, organizations, locations, products, events, or miscellaneous. The Lassy Project automatically annotated the SoNaR-1 corpus and then manually checked the results. Obtaining, annotating an evaluating data is a time-consuming process, thus showing how much effort it takes to create a proper dataset. Developing automated fine-grained multi-label named entity classification systems can help expand existing corpus with additional information to improve multi-classification. Other NLP fields improve as well for named entity classification reveals semantic information useful in different types of prediction models like text classification or QA-systems.

One other large corpus is that of Wikipedia, which is one of the most significant voluntary projects in the world. Jiang, Bai, Zhang, and Hu (2017) describes Wikipedia as a highly structured large domain-independent encyclopedic multilingual database that holds over five million articles. All articles arrange in alphabetical order, and contextual information holds internal cross-references to other articles, external references to academic literature, and groupings based on learning categories constructed semi-automatically. Each article describes a single concept, and there is only one article for each concept; thus, Jiang et al. (2017) states the title of an article is a concise sentence resembling a term of a thesaurus. However, they also note that one article can have multiple categories, thus creating an acyclic graph. According to Wikimedia Foundation (2019), this means it is not possible to

create a strict hierarchy. To overcome this problem, Wikipedia introduced the Main Topic Classifications (MTC). Every one of these classifications holds subcategories that hold information about a specific topic within the MTC, and thus an MTC could be seen as a hypernym of the hyponym subcategories it holds. Therefore, one could state that the Wikipedia MTC articles hold valuable information about named entities that useful for training a named entity classification system. Nevertheless, using MTC articles has not been researched by other scientists yet. Research involving MTC usage as proposed has yet to be conducted which leads to the question: "Can Wikipedia MTC be used to create an efficient, accurate, fine-grained multi-label named entity classification system for the Dutch language?" and if so: "Can it label entities it has not been trained on to classify?".

This study presents a multi-label classification approach that automatically retrieves data from Wikipedia and trains a named entity classification system based upon MTC and their subcategories. The hypothesis is that articles within the Wikipedia MTC are useful to develop an efficient and accurate dataset to create a new fine-grained multi-label system. By using the Wikipedia MTC information as data for the classification system information retrieval and annotation can be automated, and revealing undiscovered research territory. The development of a new fine-grained multi-label named entity classification system can help improve existing datasets by adding new generalized labels for unknown named entities such as entities that wear the label miscellaneous. This system will focus itself on the Dutch language, so the entities of the SoNaR1-corpus holding the MISC label are useful in the evaluation.

2 RELATED WORKS

This section discusses the background in the field of data mining, multi-label classification, and named entity classification. Topics like the DBpedia, Flair library, GloVe embeddings and BERT embeddings are examined as well.

2.1 BACKGROUND

A study by Zhong, Du, and Gao (2018) used involves fine-grained named entity recognition for a question and answering texts. They state that due to the architecture of QA, finding enough context information is challenging. By using a supervision DBpedia based model, they extend the information of an entity mention in a question. However, they found this was still not enough to classify an entity type correctly. Therefore, they present the entity mention to a search engine and retrieve the first ten snippets to extend the context of an entity.

They build an entity mention model that allowed them to detect related sentences of the entity mentions. By using a "data-driven greedy n-grams algorithm", it was possible to filter low-quality entity mention sentences. Combined with a k-nearest neighbour algorithm, they obtained more fine-grained entity types. They selected 5000 questions from the Text Retrieval Conference 1999-2007 QA dataset and annotated the questions regarding DBpedia's ontology classes. To construct a baseline, they used the Stanford NER. The baseline has an F1-score of 0.73 for entity mentions and 0.71 for entity types. Without an n-gram filter, the model has an F1-score of 0.68 for entity mentions and 0.62 for entity types. The model with the n-gram filter has an F1-score of 0.79 for entity mentions and 0.74 for entity types, thus proving the n-gram filter improves their system.

Compared to the Stanford NER, their system is performing better but not by much. Their system reliance on related sentences is high and therefore, not robust. Also, they do not state if the inclusion of advertisement snippets are within the first ten snippets. However, they did use a different approach to extend their context information for entity mentions compared to the previously mentioned studies. They only slightly improved on existing systems while adding extra information, and therefore one must conclude that having correct and abundant context information is crucial for the fine-grained named entity recognition field.

Lal and Chowdary (2019) state in their study, they do not make use of knowledgebases for these take time to update and because of their size are also resource-intensive. Instead, they presented SANE 2.0 that uses a parsed Wikipedia XML dump with the Dizzy Logic Wiki Parser. Parsing enabled them to extract topics with corresponding Wikipedia categories to create a database. This database holds "6,742,064 articles and 27,347,879 categories corresponding to these articles" according to Lal and Chowdary (2019). However, for the classification of locations, they created a database from Geonames, five consisting of 202 countries and their aliases, 3893 states, and 41,023 cities. They found this had better results than the categorizations by Wikipedia because the names of locations rarely change. To identify named entities they make use of the Stanford NER. When mentioning a named entity, other entity mentions in the sentence link to that entity as their type.

Otherwise, they look up the entity within their database. When finding a named entity, Lal and Chowdary (2019) put the corresponding categories in a category list and then the best five categories are selected using Word2Vec highest average similarity scores. They map the selected categories to proper Wordnet equivalents.

They are linking the found Wordnet type as a label to a named entity. The system also looks at the context of an entity by using the similarity model. If the system still fails, it makes use of the Yahoo search engine similar to the approach presented by Zhong et al. (2018). The webpage gets parsed using the Beautiful Soup Python library. After this, it selects the description part within the search results that in most cases, hold the type of the named entity according to Lal and Chowdary (2019). Then it obtains the nouns from the description, constructs a list excluding stop words, and is then send to the type selection phase. The performance of the system is evaluated by using precision and compared with the FINET system. For coarse-grained entity recognition, the FINET and SANE 2.0 systems have a precision of 96.86. For the fine-grained entity recognition, the FINET system has a precision of 80.87 and SANE 2.0 a precision of 81.49, thus outperforming the baseline.

Lal and Chowdary (2019) states they do not use a knowledge base, but this is disputable. Using a Wikipedia XML dump as data and converting it into a database from which one can extract data is almost the same as extracting information from DBpedia. The only difference lies in the way information is extracted and who can use the database. Also, both DBpedia and constructed databases need maintenance. Furthermore, SANE 2.0 uses the Yahoo search engine as one of the last resorts for finding an entity type whereas Zhong et al. (2018) study focuses on this implementation, implying using a search engine is not the best option. However, Lal and Chowdary (2019) does state it does not include advertisement snippets. The system introduced by Lal and Chowdary (2019) has a high precision accuracy and is robust due to the many steps when failing to find an entity type.

2.2 THE FLAIR LIBRARY

Flair is a library build upon Pytorch NLP framework that allows easy application of state-of-the-art NLP models and is created by Akbik, Blythe, and Vollgraf (2018). Due to the widespread Flair community, the library is also multilingual. By using Flair, one can easily create models using Flair, BERT, ELMo, or other kinds of embeddings, making it user friendly. Also, the Flair library auto-detects if the data is a multi-label classification problem. Even though the Flair library already has set default parameters, it is possible to adapt them to the specific needs for a task making it also flexible. It also allows someone to stack different word and document embeddings on top of each other, making the Flair library ideal for trying different sorts of methods for multi-label classification.

2.3 GLOVE EMBEDDINGS

Using an unsupervised learning algorithm for obtaining vector representations for words Pennington, Socher, and Manning (2014) created the pre-trained GloVe embeddings. They state the following: "Training is performed on aggregated global word-word co-occurrence statistics from a corpus, and the resulting representations showcase interesting linear substructures of the word vector space.". By doing so, they have created word space vectors that hold all probabilities for a specific instance. It takes in the whole sentence as the context for a specific word and creates a vector for this specific word summarizing the information to a statistical math equation from which a statistical model can learn. Using word GloVe embeddings will allow one to use a whole sentence as input and turning it into a vector giving every word entity with the same variable name the same vector. This method proves usefulness to convert the general retrieved information of entities to machine-readable input for a new fine-grained multi-label named entity classification system.

2.4 BERT EMBEDDINGS

BERT embeddings are similar to the GloVe embeddings but have different processing steps. By implementing BERT embeddings, the texts will get tokenized by a byte pair encoding algorithm based upon Wordpiece embeddings that lie within the BERT embeddings according to Devlin, Chang, Lee, and Toutanova (2018). When texts get processed using BERT embeddings, it will replace parts of a word with hashtags until it finds a word vector that is similar to the word it is processing. BERT embeddings also take into account specific parts of context per word, meaning it can, for example, create three vectors with the same name for one sentence but with other contexts. Therefore BERT embeddings differ from the Glove embeddings that can only hold one vector name for one sentence. BERT embeddings have a multilingual base and useful in all languages. However, there is also one Dutch variant of BERT embeddings called BERTje developed by Vries, Cranenburgh, Bisazza, Caselli, Noord, and Nissim (2019). In their study, they developed and evaluated a monolingual Dutch BERT model using the same architecture and parameters as that of the BERT model. The Dutch model only includes Dutch-based Wikipedia text and thus based upon a sizeable, diverse dataset consisting out of 2,4 billion tokens. Their study shows it outperforms the equally-sized multilingual BERT model on different sorts of natural language processing tasks for the Dutch language.

3 DATA COLLECTION AND ANNOTATION

This chapter will give an overview of how the data is collected, annotated, and processed.

3.1 COLLECTING DATA FROM MTCS

To collect the MTC data for the classification system, one must first connect to the semantic web. There are different sorts of connecting to the semantic web, but one way to do this is by making use of DBpedia, according to Mika, Ciaramita, Zaragoza, and Atserias (2008). This a lightweight ontology that makes it possible to extract data out of Wikipedia information blocks. DBpedia stores its data in RDF, which focusses on a targeted, labelled graph data format for displaying information on the web that holds metadata. DBpedia provides the ability to retrieve data using SPARQL queries by introducing at least two different concepts, namely, resource and property. The resource tells the API which item it should locate and looks like the following URL: `http://dbpedia.org/resource/ARTICLE` while a property introduces labels for the stored metadata. DBpedia is a kind of database that manages information from every article of Wikipedia. For instance, it assumes that DBpedia sees each article as a unique named entity, thus as a resource. Each resource contains a series of properties which in turn contain values. Mika et al. (2008) mainly uses DBpedia to demonstrate how human efforts are useful to annotate user-generated content to provide improved support for machine-powered annotations of residual content. They also investigate how named entity recognition technology can be used to fill infoboxes from Wikipedia automatically. They try not to classify named entities, but rather to automatically supplement Wikipedia articles linked to these named entities with information.

According to Bizer, Lehmann, Kobilarov, Auer, Becker, Cyganiak, and Hellmann (2009), DBpedia is "an effort by the community to extract structured information from Wikipedia and make it available on the Internet." They say that it contains more than 2,600,000 named entities with each having a unique identification code. Different values can be retrieved per named entity using the given properties of that resource. Bizer et al. (2009) mainly describes how DBpedia works and what the possibilities are. They argue that DBpedia consists of various information-providing components, such as the abstract. Each resource contains an abstract consisting of a maximum of five hundred words, and if one wants to retrieve this summary one uses with the *dbpedia:abstract* property.

Prud'hommeaux and Seaborne (2008) states that DBpedia provides the ability to retrieve data using the SPARQL Protocol And RDF Query Language (SPARQL). This specification defines the syntax and semantics of the SPARQL query language for RDF. SPARQL can be used to express queries in different data sources, regardless of whether the data was initially saved as RDF or viewed as RDF through middleware. SPARQL includes options for querying required and optional chart patterns along with their conjunctions and disjunctions. SPARQL also supports stretchable testing values and limiting queries on the source of the RDF graph. The results of SPARQL queries can be result sets or RDF charts, according to Prud'hommeaux and Seaborne (2008). By retrieving data using SPARQL-queries out of the DBpedia database, it is possible to retrieve abstracts per article that proves useful as training data for it holds context information of named entities.

This research uses four different properties to collect the data provided by DBpedia , namely *∧skos:broader*, *is dct:subject of*, *dct:subject*, and *dbo:abstract*. By using *∧skos:broader* one can retrieve all subcategories beneath the super-category entity and thus be able to retrieve a clear hierarchical information tree. When using *is dct:subject of* it is possible to retrieve information about an entity that falls under such a subcategory. When using *dct:subject* it works the other way around, and one can get information about a super-category that in turn might hold other entities. When finding a new entity, it is possible to extract the abstract by using the *dbo:abstract* property. To create an evenly balanced out dataset, the one must collect enough data. Four strategies will be applied to do so as can be seen in figure 1.

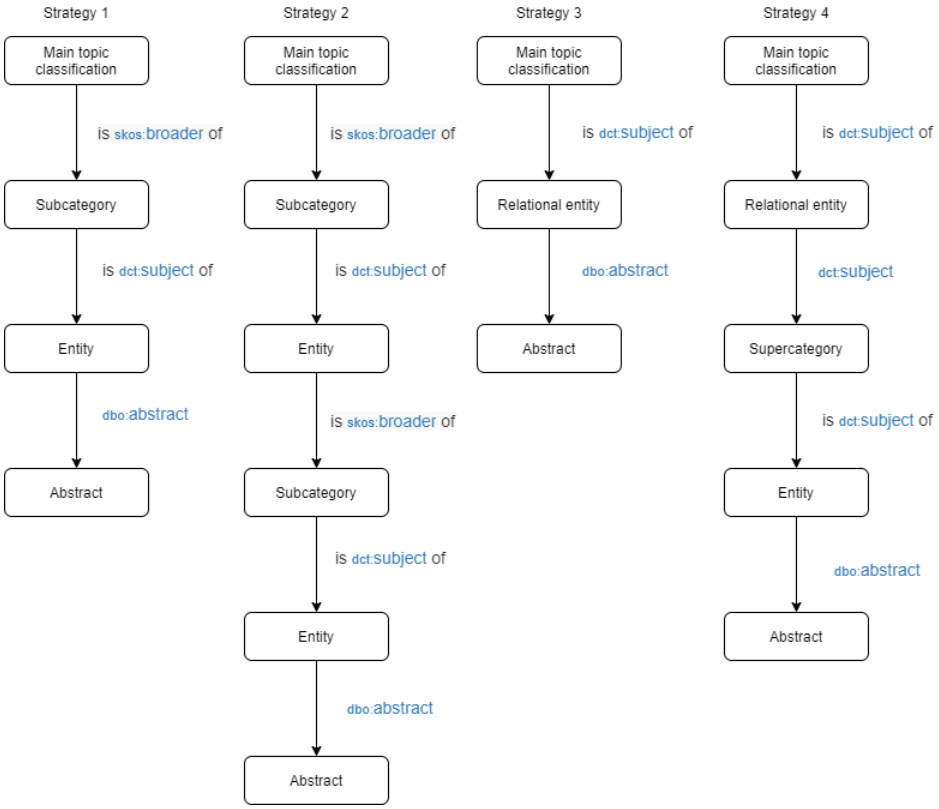


Figure 1: Collection strategies using DBpedia properties.

3.1.1 Abstract

In the example below, one can see an example of the English abstract for 'Multi-label classification'. As can be seen, the abstract is full of useful information about the entity. It consists of multiple sentences that try to explain what the entity is all about. The sentences hold different related entities of the entity 'Multi-label classification' that giving the reader a better insight into what 'Multi-label classification' means. Therefore, one could state that this information is useful for classifying named entities.

```
EXAMPLE ABSTRACT:

"In machine learning, multi-label classification and the strongly related
the problem of multi-output classification are variants of the classification
the problem where multiple target labels must be assigned to each instance.
Multi-label classification should not be confused with multi-class
```

```
classification, which is the problem of categorizing instances into one
of more than two classes. Formally, multi-label learning can be phrased
as the problem of finding a model that maps inputs x to binary vectors
y, rather than scalar outputs as in the ordinary classification problem.
There are two main methods for tackling the multi-label classification
problem: problem transformation methods and algorithm adaptation methods.
Problem transformation methods transform the multi-label problem into a
set of binary classification problems, which can then be handled using
single-class classifiers. Algorithm adaptation methods adapt the algorithms
to directly perform multi-label classification. In other words, rather
than trying to convert the problem to a simpler problem, they try to address
the problem in its full form."
```

3.1.2 Collecting strategy 1

As can be seen in figure 1, the first strategy consists of three steps. Step one looks for subcategories that fall under the MTC. Step two consists out of retrieving entities that fall under these subcategories. Step three tries to collect all abstracts of the retrieved entities. This strategy is the purest data retrieval method for it stays under the scope of the MTC branch and is the closest to the MTC topic.

3.1.3 Collecting strategy 2

The second strategy is more expansive, for it repeats steps one and two of the first strategy before collecting the abstract. By doing so, this strategy goes one level further down the three to look for new entities while still falling under the scope of the MTC topic meaning the data is still pure. However, some entities might not have a broader scope meaning some entities are the ending of a branch, and therefore the effectiveness of this strategy can be slim.

3.1.4 Collecting strategy 3

The first two strategies try to find subcategories of the MTC so that it can retain the pureness of the data. However, when the frequency of obtained entities is low one must look for a different solution. As Jiang et al. (2017) mentioned in their research, one article can have multiple categories, thus creating an acyclic graph, and therefore a strict hierarchy could not be created according to Wikimedia Foundation (2019), so the MTCs were introduced. Even though the MTCs might hold a more strict hierarchy, it still falls under the laws of Wikipedia; thus, an MTC also falls within the acyclic graph structure meaning it can also hold a relational entity. Such an entity is related to the MTC but not necessary a super- or subcategory of the MTC. However, this is strategy is less pure than the first two strategies for it does not look under the scope of the MTC, but it might be useful to retrieve enough information. Therefore this strategy tries to finds a relational entity of the MTC in step one and obtains the abstract in step two.

3.1.5 Collecting strategy 4

Strategy four is similar to that of strategy three but then more expansive for it tries to find new entities within the super-categories of the relational entities. Step one tries to retrieve the relational entities and is followed by step two that tries to retrieve a super-category of this entity. Step three then tries to locate other entities that fall under the same super-category of the relational entity, while step four tries to obtain the abstracts of these newly obtained entities. However, one must note that this strategy is the least pure for it takes in relational super-categories subjects in

order to obtain more data. There is a slight chance this strategy retrieves unrelated and overlapping information.

3.1.6 Collection review

To get a good idea of the size and distribution of the obtained entity information one can take a look at graph A.1 in the appendix. This graph shows the data collection of the first strategy for Dutch and English so one can also see the depth of Dutch representation in DBpedia compared to a world language. As can be seen, the English frequency far exceeds that of the Dutch obtained data, but the Dutch dataset is balanced more. For the English language, the MTC label Business holds the most entity abstracts with 3915 while Policy only has 145, thus making dataset skewed. For the Dutch language, the MTC label Human behavior holds the most entity abstracts with 555 and while the Policy label yet again has the lowest amount of entities with 26 entity abstracts and therefor also skewed. One must also note that the MTC Universe did not hold any categories in both languages and thus not included in this research. Compared to the English frequency, the Dutch frequency has a relatively good representation within the MTC hierarchy considering, the English language is spoken all over the world, and Dutch is not. The total Dutch frequency is 8538, and the total English frequency is 48800, as shown in table A.5 and A.6. Meaning that the Dutch language representation is still 17,50% compared to that of the English language. However, the Dutch still hold a low absolute frequency for most MTCs, and therefore one must conclude that it is wise to implement strategy two. Wikipedia itself claims the Dutch Wikipedia holds 2,027,982 articles[1] and the English Wikipedia 6,134,554 articles[2]. Compared with these frequencies, the Dutch MTC articles are only 0,004% of the total available articles and the English MTC articles only 0,008%. Meaning not all articles that could fall under the MTCs can be extracted.

To see the impact of the second strategy, one can look at graph A.2 in the appendix. As can be seen in the graph, the overall frequency of data has not increased; thus, one must conclude that the second strategy has failed. More data is needed, and so strategy three must be conducted. Strategy three also shows a minor increase in the overall frequency when examining A.3 in the appendix. This extra information should be less pure for it now looks at entities that are related to the same level as the MTCs. However, there are still low frequencies for the MTC Policy with only 31 entities and MTC Object with only 41 entities. Having enough data is crucial, and therefore the last strategy must also be conducted even though this might retrieve unrelated and overlapping data. When examing A.7 in the appendix, one can see there is a significant increase in all frequencies with outliers in the MTCs Education, Human behavior, Language, Technology and a couple more. As expected, this strategy has retrieved the most information for the MTCs. At first glance, some MTCs benefit from this strategy. For instance, MTC Politics has some overlap with the MTCs Government, History, Crime, and Science. One must consider this overlap is acceptable for there are few new entities, and these topics are included in Politics as well. However, when examing the MTC Religion, it stands out that there is a lot of birth and death year from or before the 17th-century categories that hold many entities about people. Some of these people may not have a proper link to Religion. MTC Science also holds birth year categories that are related to people considered not linked to Science. It also holds the abstracts of some other MTCs categories. When considering the MTCs Science and Technology, one can see it also holds similar birth and death of people properties. These people hold a wide variety of occupations like athletes, actors, writers, politicians, and so on. One can state that strategy four does retrieve more information but also causes a substantial

[1] https://en.wikipedia.org/wiki/Dutch_Wikipedia
[2] https://en.wikipedia.org/wiki/English_Wikipedia

amount of noise in the dataset. Considering that strategy three retrieves the purest data with the least amount of noise, one must conclude that strategy three is the best data collection strategy.

3.1.7 Overlapping MTCs review

After automatically retrieving the abstracts of entities, it is important to consider the architecture of the data. As can be seen in graph A.1 situated in the appendix, the dataset is unbalanced, and thus a decision must be made to even things out. Before doing this, it is wise to see if the MTCs have any connections with one and another. As Jiang et al. (2017) stated in their study, Wikipedia is an acyclic graph implying some of these MTCs might overlap with one and other. To see if any MTC has overlap the property skos : broader is checked by hand to examine if the value of this property is one of the other MTCs. These detected patterns are documented in figure 2.

Sixteen MTCs are not linked to any of the other MTCs and thus are the purest labels to work with for no overlap is detected. These sixteen labels are Concepts, Crime, Education, Geography, Human behavior, Industry, Knowledge, Mass media, Mathematics, Military, Mind, Objects, Organisations, People, Sports, and World. When looking at the MTCs that link, one can see five main MTCs hold a higher status over the other MTCs, namely: Science, Health, Society, Nature, and Technology. From these five MTCs, only Health, Nature, and Technology have unique MTCs below them that are unconnected to one of the five MTCs. Therefore, one can conclude it is safe to merge Food and drink within the MTC label Health, Life, and Energy into the MTC label Nature and Engineering into the MTC label Technology to increase the frequency of entities for this MTC label. When examining the MTCs Science and Society, one can see these MTC label tree structures connect by Humanities.

3.1.8 Annotation strategies

When increasing the frequency of entities per MTC label, the dataset gets even more unbalanced, leading to unrealistic results. When balancing the dataset, there are three options to consider. The first option being, setting a maximum on retrieving entities to the total amount of entities retrievable for the MTC label that holds the lowest frequency of entities. Doing so would result in a maximum of 31 entities per MTC label for this is the total amount of the Policy label. However, this option provides little data to create a robust system. As a second option, one can take MTCs out of the equation, or merge MTCs. However, merging labels can only be conducted if the label has only one super-category. Because Science and Humanities link with Humanities every MTC beneath these MTCs cannot merge. Also merging Food and drink into health might not be needed for it holds an average frequency size of entities. The third option would be setting a range for the frequency of entities an MTC must hold for accepting it as a participant in the experiment. By setting a minimum and maximum frequency of entities threshold for an MTC label to participate, one could reduce the number of labels without having to retrieve extra unrelated and overlapping information. In the study done by Akoka, Wattiau, du Mouza, Fadili, Lammari, Métais, Cherfi, and Cherfi (2014), they determined if data was sensitive by using the standard deviation to determine a range. By using the standard deviation, one can determine the margin of error to set a range threshold to determine if the frequency of entities within an MTC is too low or too high. This strategy will give an insight into which MTC should be removed, merged, or pruned and is therefore considered the best strategy.

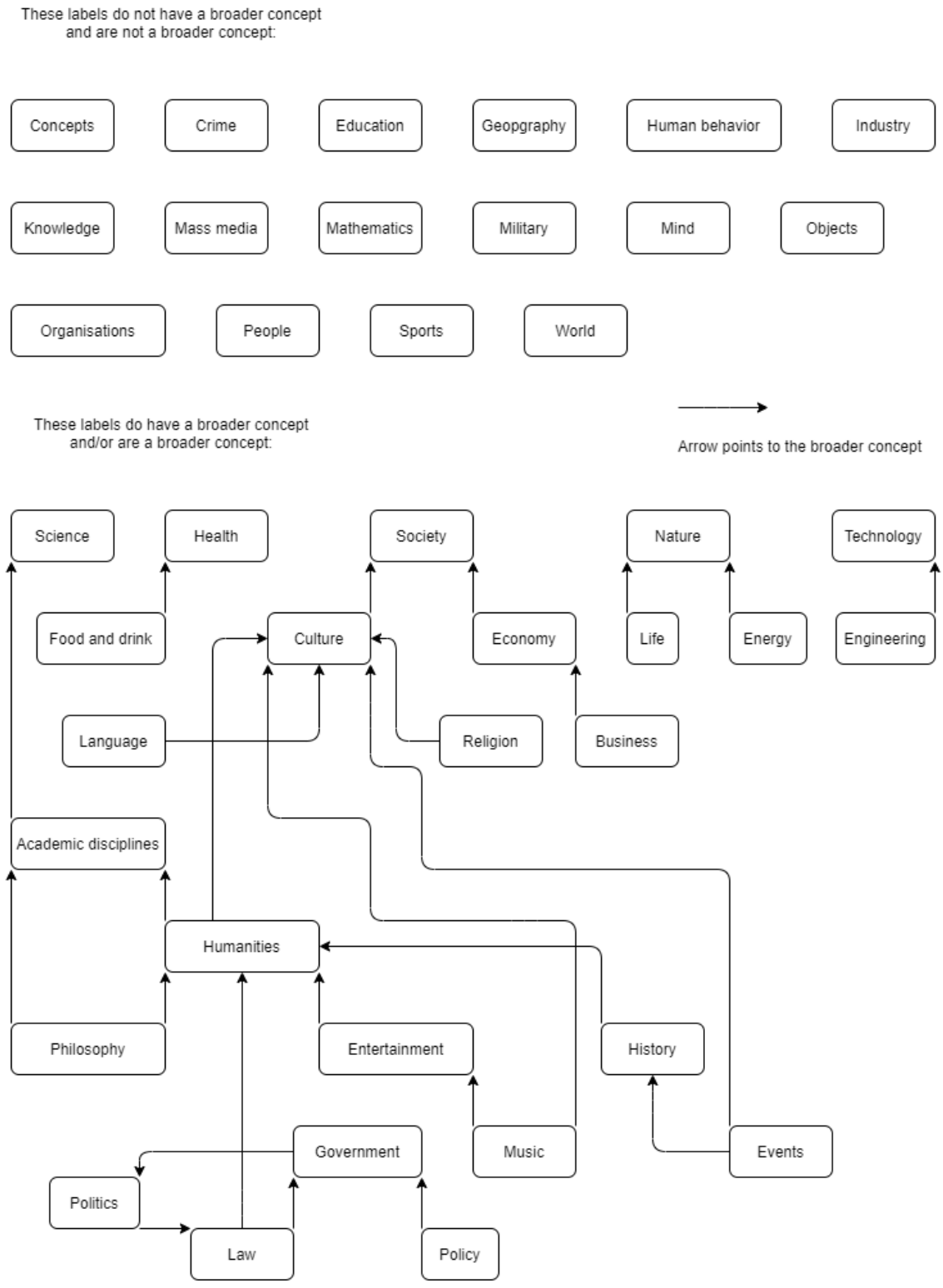


Figure 2: Acyclic graph of Wikipedia's MTCs.

3.1.9 Calculating the range threshold

According to Levshina (2015), the standard deviation is a calculation used to determine how to spread out measurements are from the average. A high standard deviation states that measurements spread out more and a low standard deviation states a measurement is very close to the average. In equation 1, one can see the formula for calculating the standard deviation. The x_i symbol stands for the fre-

quency of entities an MTC holds, the $\bar{x}$ stands for the mean value of the data set, and N is the total frequency of entities in a dataset. Equation 2 holds the formula for the mean value. The frequency of entities per MTC label retrieved using the third collection strategy, as seen in graph A.3 in the appendix. In table 1, one can see the results of this formula when used upon dataset retrieved using collection strategy three. The margin of error is either one standard deviation below or above the mean of the total dataset thus ideal for determining the range threshold for the frequency of entities that an MTC holds.

$$s = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \bar{x})^2} \tag{1}$$

$$\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i \tag{2}$$

Formula	Result
Sum frequency entities	8937
Sum MTCs	41
Mean	218
Variance	19287
Standard deviation	139
Margin of error	79 - 357

Table 1: Formulas needed to set threshold range being the margin of error.

3.1.10 Label allocation

Now the range threshold has been set to 79 - 357; one can execute three strategies to allocate proper labels to the data, namely by removing, merging, or pruning data per MTC. When looking at graph A.3 in the appendix one can see the frequencies of entities per MTC and thus can conclude which MTC must undergo which method. The first step is checking which MTCs fall within this range which as seen in table 9 in the appendix. The second step is pruning the MTCs that have a higher frequency of entities than the threshold permits, and these results display themselves in table 10 in the appendix. A total of 629 entities have been pruned with the MTC Human behavior losing the most entities with 238. The third step, one has to identify the entities that fall under the range because these MTCs have to be removed or merged with another MTC. In the appendix table 11, shows that 394 entities potentially have to be removed and a total of 159 was needed to include them in the test. From these MTCs Concepts, Mind, Objects, Organizations, and People do not belong under the other MTC according to figure 2. Thus are removed out of the equation. However, the MTC Events has 77 entities and falls under MTCs History and Culture. Culture is already pruned to 357 entities and thus can not merge with Events. History only holds 239 entities and if merged with Events has 316 entities, thus still within range. The MTC Policy falls under the pruned Government classification, and therefor Policy will also be removed out of the equation. Meaning 5390 entities are within range, and one has to remove 946 entities and merge 77 entities. Therefore, the dataset will contain 7991 entities, and 34 MTC labels, as can be seen in table 12 in the appendix. One has to note that MTC labels such as Events, Organizations, and People will be underrepresented or non-existing in the created dataset. However, in the SoNaR1-corpus, this is the opposite for they are. These are three of the highest classifications given to entities in this corpus.

3.1.11 Processing abstracts

All abstracts retrieved are stripped of unnecessary characters using regex. Doing so removes summarizing punctuation and other non-informative units. Keeping stop words, and casing might be useful to learn context positioning and semantic relations.

```
Regex: [^\.a-zA-Z0-9]
```

3.2 COLLECTING DATA FROM SONAR1-CORPUS

This section discusses how the data collection, processing, and annotation occurs for the SoNaR1-corpus.

3.2.1 Formatting

This study will use pre-processed files created by Jonas Schuitemaker[3] to retrieve MISC named entities. Jonas has restructured the text of the SoNaR-1 dataset. In doing so, he has maintained the following distribution:

Line-ID *TAB* **Format-ID** *TAB* **Word** *TAB* **Category** *TAB* **Subcategory**

```
DOCUMENT    wiki7793_words.xml
...
38  38.0    De
39  39.0    totale
40  40.0    lengte
41  41.0    bedraagt
42  42.0    78
43  43.0    kilometer
44  44.0    ,
45  45.0    waarvan
46  46.0    45
47  47.0    in
48  48.0    Belgie  LOC LAND
49  49.0    .
SENTENCE    De totale lengte bedraagt 78 kilometer , waarvan 45 in Belgie .
```

The files also indicate when a document has started and display entire sentences on one line. Extracting a MISC named entity can be done by retrieving the fourth element of a sentence, and checking if it contains MISC. Jonas then split the formatted data into training, development, and test sets. Obtaining these text files can be done by visiting his GitHub[4].

3.2.2 Annotation

It is valuable to test the reliability of alternative classifications. Also, the annotation results provide a measurement tool for evaluating a multi-label classification system. For this purpose, annotating the test set of the formatted data by two objective external annotators is necessary. The annotators must able to classify the named entities in one of the 34 proposed MTC labels. For this, creating a text format that holds 100 named entities is needed. Each sentence in this text contains a named entity with a MISC classification. Distributing this text file is necessary to find multiple objective annotators.

[3] https://linkedin.com/in/jonasschui
[4] https://github.com/jonasschui/SoNaR_corpus_formatted

3.3 EVALUATING MANUAL ANNOTATIONS

Two annotators have been able to annotate the MISC named entities of the SoNaR1-corpus using a prepared text. Both annotators annotated the text file resulting in two annotated text files creating a new text file containing the gold standard using the agreed-upon labels meaning the union of both sets of labels. Due to the multi-label classification task, an agreement exists if both annotators have assigned at least one label. Using the gold standard to calculate the accuracy, precision, recall, and macro F1-score one can evaluate the predictions of the best model[5]. These formulas are presented in chapter 5.

3.3.1 Kappa-formula

The two annotators must agree before one can execute the evaluation process. This study includes a total of 100 named entities. Ensuring the validity and reliability of these annotations, one can use the Kappa formula developed by Cohen (1960). The Kappa formula is an intra- and interobserver agreement where the Kappa coefficient is a probability-adjusted measure of agreement between assessments. A Kappa-score of zero means that the agreement between annotators based on chance, a Kappa-score of one is a complete agreement. The ranges go from:

```
0 No agreement
0 - .20 Slight
.21 - .40 Fair
.41 - .60 Moderate
.61 - .80 Substantial
.81 - 1.0 Perfect
```

Therefore, the Kappa formula demonstrates whether alternative classifications are usable. The letter A stands for the number of matches and the letter E for the probability of matches. In equation 3 one can see the Kappa-formula.

$$\text{Kappa} = \frac{A - E}{1 - E} \tag{3}$$

3.4 RESULTS OF KAPPA-FORMULA

Both annotators agreed on 82 of the 100 MISC entities, and therefore the intra- and interobserver agreement has a Kappa score of 0.8182. The high Kappa score implies that the alternative classifications are useful for annotators because it is possible to annotate consistently. Meaning the annotators were in perfect agreement, and thus the manual annotations can be used as the gold standard. The gold standard acts as a measurement tool to evaluate the accuracy of the system that automatically assigns classifications.

[5] All annotations, including the gold standard of this research, are at: "https://github.com/richardscholtens/Master_Thesis".

4 METHOD

In this section, the methods conducted will be discussed. A baseline is created by implementing the DummyClassifier from SciKit Learn developed by Pedregosa, Varoquaux, Gramfort, Michel, Thirion, Grisel, Blondel, Prettenhofer, Weiss, Dubourg, Vanderplas, Passos, Cournapeau, Brucher, Perrot, and Duchesnay (2011). First, a multi-label text classification method is discussed using the Flair framework developed by Akbik, Blythe, and Vollgraf (2018). For this, the retrieved dataset is split into a training set of 60%, a development set of 20%, and a test set of 20%. The second method uses the text classification models to perform a fine-grained multi-label named entity classification.

4.1 DUMMYCLASSIFIER BASELINE

To evaluate how well the system performed a DummyClassifier baseline was introduced. This algorithm appoints the most frequent label to all predictions meaning that the gold standard always compares with only one label prediction. Such a method helps to set a threshold to evaluate how well other models perform.

4.2 FLAIR LSTM NEURAL NETWORK

Flair is a state-of-the-art Natural Language Processing framework which allows binary, multi-class, and multi-label classification using neural networks like an RNN or an LSTM. It also allows one to quickly implement different kinds of word embeddings such as Flair, ELMo, Glove, and BERT Embeddings. However, this study will limit itself to the use of a bidirectional LSTM neural network using pre-trained Glove[1], and BERTje embeddings to tokenize the texts[2]. By using the TextClassifier class within the Flair library[3] it is possible to use the whole abstract as training input. One must note that the processing for BERTje embeddings is different for it can only take in 512 tokens; thus, splitting the input in chunks of 200 tokens for both types of embeddings. By doing so, one can also increase the frequency of input features, thus resulting in more pure and specific data. By chunking, an input text one does not necessarily have to begin with the start of a sentence. Using a bidirectional model will help to learn more information for it takes context before and after an entity in consideration. An LSTM neural network remembers information retrieved from earlier layers and thus will give more robust results. The neural networks were first tested on the development set, then on the test set. It was first run with the default parameters, a learning rate of 0.001 and one epoch as can be seen in table 2. After this, the models were fine-tuned by examining the learning rate graph to see which learning rate would be best to train on and setting the frequency of epoch to ten. To see how well fine-tuning was done, the training loss was compared with the validation loss. By doing so, one could evaluate if the model was under fitted, over fitted or good fitted.

[1] GloVe, URL: https://github.com/stanfordnlp/GloVe
[2] BERTje, URL: https://github.com/wietsedv/bertje
[3] flairNLP, version 0.4.5, URL: https://github.com/flairNLP/flair

Parameter	Value
learning_rate	0.001
mini_batch_size	64
patience	3
anneal_factor	0.5
max_epochs	1
shuffle	True
train_with_dev	True
batch_growth_annealing	False
bidirectional	True
dropout	0.5
loss_function	BCEWithLogitsLoss()

Table 2: Default parameters set for models without fine-tuning.

4.2.1 Learning rate graph

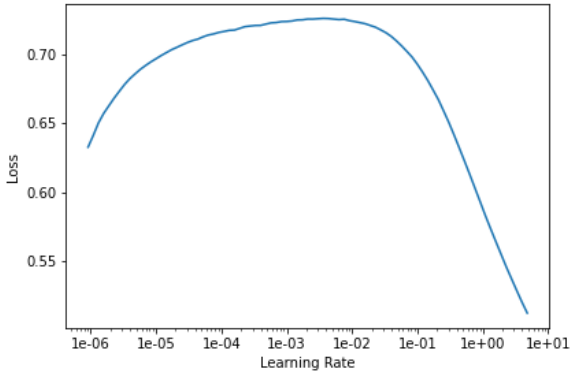


Figure 3: Learning rate of LSTM model using BERTje embeddings

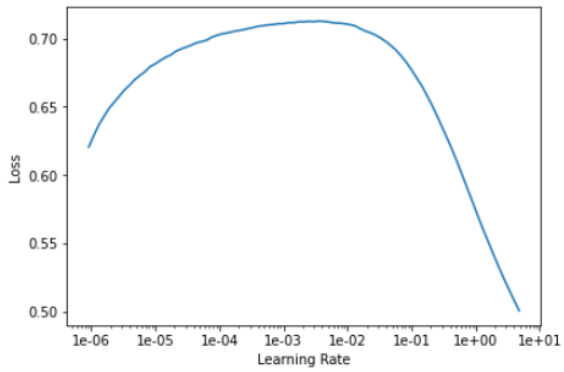


Figure 4: Learning rate of LSTM model using GloVe embeddings

When examining figure 4, one can retrieve the loss for each learning rate for the best LSTM model without fine-tuning. The plot was created according to the method proposed in the study by Smith (2015). By using this method, one can find the learning rate that has the highest amount of loss. Lower learning rates mostly do not improve the loss while an optimal learning rate range shows the steepest loss drop. The final phase of the learning rate, the loss mostly increases again as the learning rate becomes too big. Using such a plot can help to select the best learning rate in the optimization process and speed up the training process. As can be seen in figure 4 and figure 4, both models have the best perfect learning rate

between 1e-03 and 1e-01; thus the learning rate was set for both models to 0.001 to retrieve the best results.

4.2.2 Classifying MISC named entities

After fine-tuning the best model, the named entity classification was started. By using the best model, it was possible to test the model on the SoNaR1-corpus by comparing the predictions of the model with the gold standard of the annotators. Therefore, it was necessary to create predictions on the MISC entities with the best-fine-tuned model.

5 EVALUATION

This section discusses the evaluation metrics used to evaluate the text classification models and the fine-grained named entity classification models.

5.1 ACCURACY, PRECISION, RECALL AND F1-SCORE

Evaluating the results of the LSTM models, accuracy, precision, recall and macro F1 score was used. Looking at other formulas besides accuracy also measures the robustness of the system. These formulas were calculated using the following terms: "True Positives (TP), True Negatives (TN), False Positives (FP) and False Negatives (FN)". Manning, Raghavan, and Schütze (2008) states that a TP result is an outcome in which the model correctly predicts the positive class. Likewise, a TN is an outcome in which the model correctly predicts the negative class. An FP result is an outcome in which the model mispredicts the positive class. An FN is an outcome in which the model mispredicts the negative class. To calculate the accuracy, precision, recall and macro F1-score, Manning et al. (2008) used the formulas as can be seen in equation 4, 5, 6, and 7.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{4}$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{5}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{6}$$

$$F_1 = 2 * \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{7}$$

5.2 ANNOTATED GOLD STANDARD

After annotating 100 MISC entities of the SoNaR1-corpus, the annotators had an intersection agreement upon 82 entities and came to an intra- and interobserver agreement Kappa-score of 0.8182. These annotations are in perfect agreement and are therefore considered the gold standard that was compared to the predictions of the best-fine-tuned LSTM model. If at least one label of the model prediction labels intersects with the annotator labels, it will be considered as a positive prediction and otherwise as a false prediction and thus a simpler version of the accuracy formula proposed in section 5.1. The accuracy formula, as proposed in section 5.1, can be influenced by a high amount of FP, and FN. Because the classification model has to choose out of 34 labels, one must acknowledge that the chances for these influences are too high and thus must be simplified. Such a prediction has to have at least a score of 0.50 to be considered a proper label. The accuracy is evaluated by comparing the frequency of total positive predictions against a total of 100 annotations.

6 RESULTS AND DISCUSSION

All results will be discussed in this chapter and are derived from LSTM neural networks using GloVe and BERTje embeddings. The baseline was conducted using the DummyClassifier.

6.1 TEXT CLASSIFICATION WITHOUT FINE-TUNING

The DummyClassifier always predicts the most frequent label and thus always chose the label Humanities on both the development as the test set. As can be seen in table 3 both LSTM models that were not fine-tuned ran on one epoch had beaten the DummyClassifier baseline with F1-scores around the 90%. Thus one can state that the models can effectively classify multi-labels to chunks of texts. When comparing the LSTM model using GloVe embeddings with the LSTM model using BERTje embeddings, one can see they have similar results. However, the LSTM model using GloVe embeddings has the highest macro F1-score of 94% and micro F1-score of 97,61% in each testing run on the test set. The LSTM using BERTje had a macro F1-score of 91,64% and a micro F1-score of 96,71% on the test set. Using GloVe embeddings on the test set resulted in an accuracy of 37,18% and using BERTje embeddings accuracy of 33,16%. It feels counter-intuitive that the F1-scores are relatively higher than the accuracy, but this can be explained for the accuracy also takes into account the amount of TN while precision and recall do not. The F1-score is calculated by precision and recall; thus, TN is not taken into this equation which can result in the significant differences when compared to the accuracy score.

Models	Accuracy	Precision	Recall	F1-score (macro)
DummyClassifier Dev	0.0424	0.0012	0.0294	0.0024
LSTM GloVe Dev	**0.3520**	**1.0000**	0.8331	0.8938
LSTM BERTje Dev	0.3202	1.0000	**0.8544**	**0.9130**
DummyClassifier Test	0.0476	0.0014	0.0294	0.0027
LSTM GloVe Test	**0.3718**	**1.0000**	**0.8949**	**0.9400**
LSTM BERTje Test	0.3316	1.0000	0.8629	0.9164

Table 3: Accuracy, precision, recall and macro F1-score per model running 1 epoch.

Models	Accuracy	Precision	Recall	F1-score (micro)
LSTM GloVe Dev	**0.3520**	**1.0000**	**0.9495**	**0.9741**
LSTM BERTje Dev	0.3202	1.0000	0.9347	0.9663
LSTM GloVe Test	**0.3718**	**1.0000**	**0.9532**	**0.9761**
LSTM BERTje Test	0.3316	1.0000	0.9363	0.9671

Table 4: Accuracy, precision, recall and micro F1-score per model running 1 epoch.

6.2 TEXT CLASSIFICATION WITH FINE-TUNING

When looking at table 5, one can see that the highest accuracy scores is reached with the LSTM model using BERTje. Both systems seemed robust enough for the results of the LSTM model using GloVe on the test set running for ten epoch had a macro F1-score of 89% and therefore only dropped 5% in macro F1-score. However, the same model had a micro F1-score of 97,60% thus had even better results. When examining the LSTM model using BERTje, one can see it resulted in a macro F1-score of 86%, thus only dropping 4%. The micro F1-score of this model one even surpassed the other scores with a result of 97,91%. Because both models are robust enough one can state accuracy is the essential factor to take in consideration. The LSTM model using BERTje embeddings retrieved the highest accuracy of 44,01% on the test set and therefore has the highest chances of succeeding in the multi-label classification task.

Models	Accuracy	Precision	Recall	F1-score (macro)
DummyClassifier Dev	0.0424	0.0012	0.0294	0.0024
LSTM GloVe Dev	**0.3657**	1.0000	**0.8536**	**0.9071**
LSTM BERTje Dev	0.0114	0.6176	0.1768	0.2427
DummyClassifier Test	0.0476	0.0014	0.0294	0.0027
LSTM GloVe Test	0.3574	1.0000	**0.8415**	**0.8987**
LSTM BERTje Test	**0.4401**	1.0000	0.8063	0.8631

Table 5: Accuracy, precision, recall and macro F1-score per model running 10 epoch.

Models	Accuracy	Precision	Recall	F1-score (micro)
LSTM GloVe Dev	**0.3657**	1.0000	**0.9522**	**0.9755**
LSTM BERTje Dev	0.0114	1.0000	0.2717	0.4274
LSTM GloVe Test	0.3574	1.0000	0.9531	0.9760
LSTM BERTje Test	**0.4401**	1.0000	**0.9590**	**0.9791**

Table 6: Accuracy, precision, recall and micro F1-score per model running 10 epoch.

6.3 EXAMINING BEST MODELS WITH FINE-TUNING

As can be seen in figure 5, the LSTM model using BERTje was still under fitted. The training and validation loss was decreasing at a similar pace and was almost horizontal. One also has to note that the lines did not touch and thus, more fine-tuning was needed. An upside was that the training data seems to hold enough information to learn from for it does not show any high spikes in the losses. However, increasing the amount of epoch would probably not have much effect for it already was near horizontal, thus altering other parameters must be considered in future research to close the gap. The same trend can be seen when examining figure 6 and therefor these fine-tuned models are still under fitted.

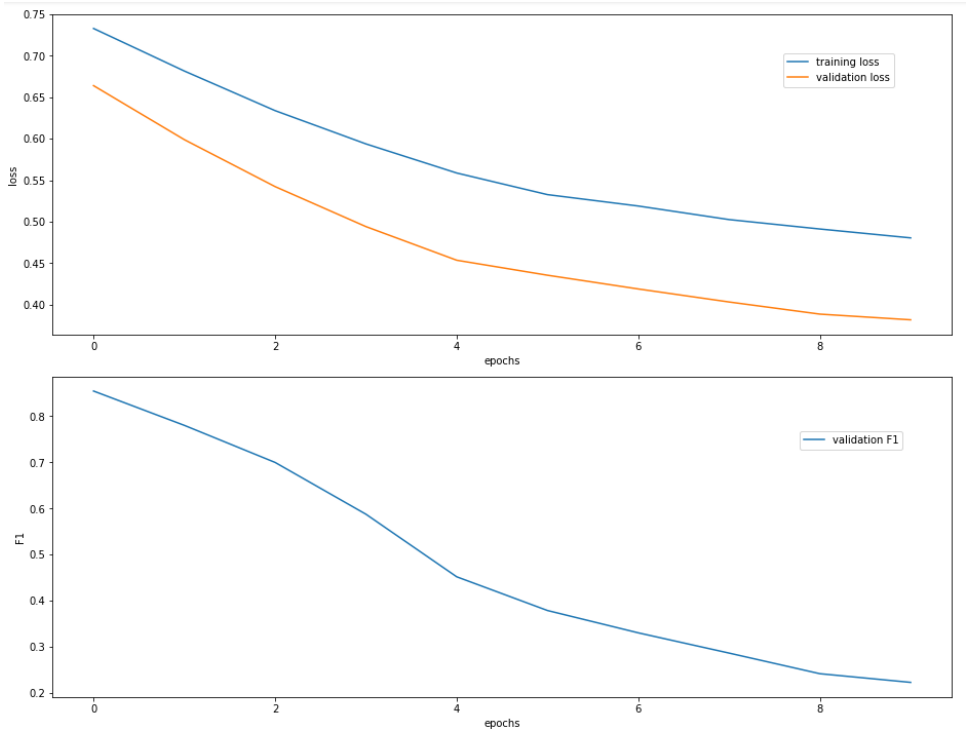


Figure 5: Training and validation loss of LSTM model using BERTje embeddings run on 10 epoch

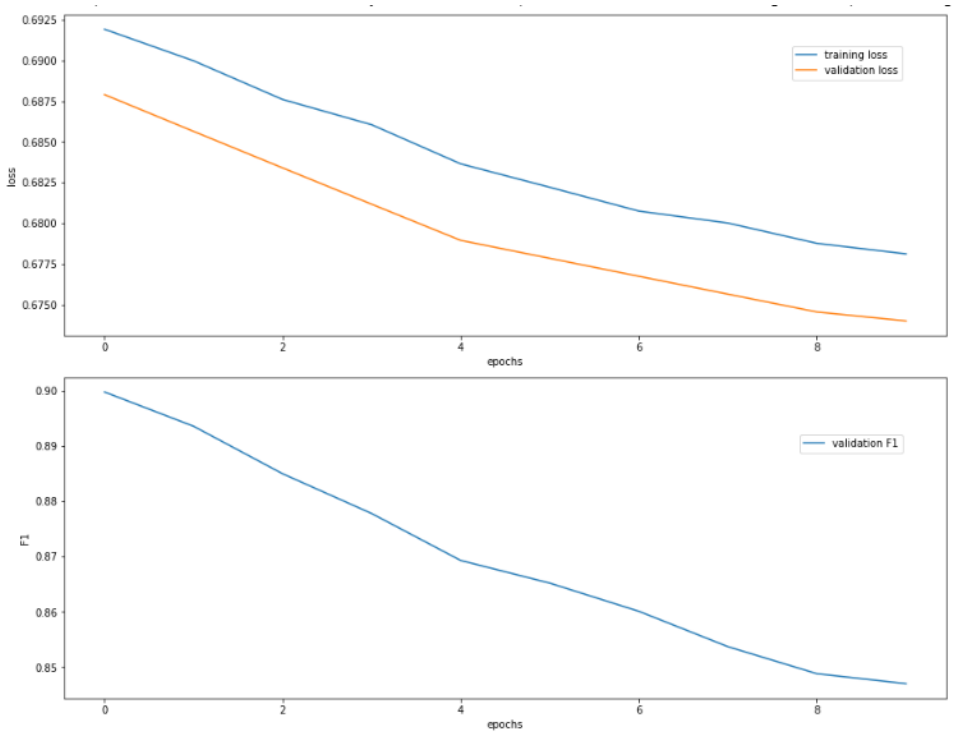


Figure 6: Training and validation loss of LSTM model using GloVe embeddings run on 10 epoch

6.4 NAMED ENTITY PREDICTIONS ON GOLD STANDARD

After fine-tuning the best LSTM model one had to see how well it would perform on an existing dataset. Therefore the models were used to make predictions on the gold standard derived from the SoNaR1-corpus. In table 7 and 8, one can see that the best-fine-tuned LSTM model using BERTje had an accuracy of 91%, macro F1-score of 8,72%, and a micro F1-score of 17,61%. The best-fine-tuned model using GloVe had an accuracy of 70%, macro F1-score of 9,26%, and a micro F1-score of 12,45%. Ultimately the baseline only has an accuracy of 8%, macro F1-score of 0,36%, and a micro F1-score of 4,26%. It seems that the baseline chose the label Humanities 8% of the time as the proper label, and thus the fine-tuned models beat the baseline with the LSTM model using BERTje being the most accurate.

Models	Accuracy %	Accuracy frequency	Precision	Recall	F1-score
DummyClassifier	0.08	8/100	0.9776	0.0244	0.0036
Best LSTM GlovE	0.70	70/100	0.3665	0.3538	**0.0926**
Best LSTM BERTje	**0.91**	**91/100**	**0.3856**	**0.3612**	0.0872

Table 7: Accuracy, precision, recall and macro F1-score of best fine-tuned models.

Models	Accuracy %	Accuracy frequency	Precision	Recall	F1-score
DummyClassifier	0.08	8/100	0.0800	0.0290	0.0426
Best LSTM GlovE	0.70	70/100	0.0753	0.3587	0.1245
Best LSTM BERTje	**0.91**	**91/100**	**0.1060**	**0.5218**	**0.1761**

Table 8: Accuracy, precision, recall and micro F1-score of best fine-tuned models.

7 CONCLUSION

This study was conducted to see if it was possible to create an efficient and accurate fine-grained named entity multi-label classification system using Wikipedia's MTCs as labels. Creating such a system will help other academics to improve existing data sets by adding new generalized labels for unclassified named entities. First data was retrieved using articles that fall under the MTC topics. The data retrieved was then used to train LSTM models to perform multi-label text and named entity classification tasks.

7.1 DATA

One had to collect information about Wikipedia's MTCs by using SPARQL-queries to retrieve the Dutch abstracts of the articles that fell under the MTCs from DBpedia. Four collection strategies were conducted to retrieve the data resulting from that the third collection strategy was the purest and had retrieved the highest frequency of data. After the first strategy was conducted, one could also state that the Dutch data obtained itself was suitable to use. However, the data distribution across each label was unbalanced, and therefor a selection of usable data had to be made to create a more balanced data set. A margin of error range threshold was set using the standard deviation. A label could only be used if it had an entity frequency that fell within the 79 - 357 entities. A total of 5390 entities were in range, 629 entities have been pruned with the MTC Human behavior losing the most entities, and the 77 entities of the MTC Events were merged with History resulting in a total of 7991 entities spread over 34 MTCs.

7.2 METHODS AND RESULTS

A baseline was set using the DummyClassifier, resulting in a baseline accuracy score of 4,76% and macro F1-score of 0,27%. When comparing the LSTM models without fine-tuning and running on one epoch, it is the LSTM using GloVe embeddings retrieved the highest accuracy with 37,18%, a macro F1-score of 94% and a micro F1-score of 97,61%. The LSTM model using BERTje embeddings retrieved an accuracy of 33,16%, macro F1-score of 91,64% and a micro F1-score of 96,71%. These fine-tuned models resulted in an accuracy of 35,74%, a macro F1-score of 89,87% and a micro F1-score of 97,60% for the LSTM model using GloVe embeddings and an accuracy of 44,01%, a macro F1-score of 86,31% and a micro F1-score of 97,91% for the LSTM model using BERTje embeddings. Both models outperformed the DummyClassifier, which had an accuracy of 4,24% and a macro F1-score of 0,24%. The fine-tuned model was then evaluated on the SoNaR1-corpus by using the texts adjusted by Jonas Schuitemaker to create a manual evaluation gold standard with the help of two annotators. To evaluate the validity and robustness of the manual annotations, the Kappa-formula was used. It was resulting in an 82 out of 100 agreements making the Kappa-score 0.8182. By Kappa-standard this is considered to be in perfect agreement thus making the 100 annotations perfect as a gold standard. A union of annotated labels were used as the gold standard, and evaluating the models resulted in an accuracy of 8%, a macro F1-score of 0,36%. The fine-tuned LSTM model using GloVe embeddings had an accuracy of 70%, a macro F1-score of

9,26%, and a micro F1-score of 12,45%. The LSTM model using BERTje embeddings an accuracy of 91%, a macro F1-score of 8,72%, and a micro F1-score of 17,61% thus proving to be the best-fine-tuned LSTM model by outperforming the baseline by 83%.

7.3 DERIVED CONCLUSIONS

One has to conclude that it is possible to create a fine-grained multi-label named entity classification system using Wikipedia's MTCs. However, one must note retrieving data in this manor must undergo specific removing, merging and pruning steps. The Dutch Wikipedia MTC articles are suitable but also limited in terms of frequency. To properly train robust neural networks using the MTC articles as training data, more information is needed. All models were still under fitted and therefore, one could argue that these results might not mean anything for the models are not robust. More fine-tuning could be done to improve the accuracy of the models. The study was conducted so that an efficient and accurate named entity classification system could be developed. However, the accuracy scores based on the test set could still be improved. On the other side, the high accuracy on the gold standard shows promise. Still, one has to consider that the gold standard consists out of 100 annotations only, and therefore one could argue more annotations are needed to conduct a proper evaluation.

7.4 FUTURE WORKS

By using English data, one can retrieve more data to work with and therefor can create a larger dataset and try to these methods for English. More Dutch data will help to increase the robustness of the models, and this might help optimize the models. Also, it might be possible to use the *is* `http://purl.org/linguistics/gold/hypernym` *of* property and create a new data retrieval strategy thus retrieving more abstracts and creating a more significant data set. Another step would be the further optimization of parameters for the best model from this study. There is still room for improvement for the best model is still under fitted.

BIBLIOGRAPHY

Alan Akbik, Duncan Blythe, and Roland Vollgraf. 2018. Contextual string embeddings for sequence labeling. In *COLING 2018, 27th International Conference on Computational Linguistics*, pages 1638–1649.

Jacky Akoka, Isabelle Wattiau, Cédric du Mouza, Hammou Fadili, Nadira Lammari, Elisabeth Métais, Samira Cherfi, and Cherfi. 2014. A semantic approach for semi-automatic detection of sensitive data. *Information Resources Management Journal*, 27:23–44.

Christian Bizer, Jens Lehmann, Georgi Kobilarov, Sören Auer, Christian Becker, Richard Cyganiak, and Sebastian Hellmann. 2009. Dbpedia - a crystallization point for the web of data. *Journal of Web Semantics*, 7(3):154 – 165. URL http://www.sciencedirect.com/science/article/pii/S1570826809000225, the Web of Data.

Ricardo Cerri, Rodrigo C. Barros, and André C.P.L.F. de Carvalho. 2014. Hierarchical multi-label classification using local neural networks. *Journal of Computer and System Sciences*, 80(1):39 – 56. URL http://www.sciencedirect.com/science/article/pii/S0022000013000718.

Jacob Cohen. 1960. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20(1):37–46. URL https://doi.org/10.1177/001316446002000104.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Yuncheng Jiang, Wen Bai, Xiaopei Zhang, and Jiaojiao Hu. 2017. Wikipedia-based information content and semantic similarity computation. *Information Processing Management*, 53(1):248 – 265. URL http://www.sciencedirect.com/science/article/pii/S0306457316303934.

Anurag Lal and Ravindranath C. Chowdary. 2019. Sane 2.0: System for fine grained named entity typing on textual data. *Engineering Applications of Artificial Intelligence*, 84:11 – 17. URL http://www.sciencedirect.com/science/article/pii/S0952197619301071.

N. Levshina. 2015. *How to do Linguistics with R: Data exploration and statistical analysis*. John Benjamins Publishing Company. URL https://books.google.nl/books?id=dGnlCgAAQBAJ.

Jin Liu, Lina Wang, Mingji Zhou, Jin Wang, and Sungyoung Lee. 2018. Fine-grained entity type classification with adaptive context. *Soft Comput.*, 22(13):4307–4318. URL https://doi-org.proxy-ub.rug.nl/10.1007/s00500-017-2963-2.

Kim Luyckx, Frederik Vaassen, Claudia Peersman, and Walter Daelemans. 2012. Fine-grained emotion detection in suicide notes: A thresholding approach to multi-label classification. *Biomedical Informatics Insights*, 5s1:BII.S8966. URL https://doi.org/10.4137/BII.S8966, pMID: 22879761.

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to Information Retrieval*. Cambridge University Press.

P. Mika, M. Ciaramita, H. Zaragoza, and J. Atserias. 2008. Learning to tag and tagging to learn: A case study on wikipedia. *IEEE Intelligent Systems*, 23(5):26–33.

Nelleke Oostdijk, Martin Reynaert, Véronique Hoste, and Ineke Schuurman. 2013. *The Construction of a 500-Million-Word Reference Corpus of Contemporary Written Dutch*, pages 219–247. Springer Berlin Heidelberg, Berlin, Heidelberg. URL https://doi.org/10.1007/978-3-642-30910-6_13.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Eric Prud'hommeaux and Andy Seaborne. 2008. SPARQL Query Language for RDF. W3C Recommendation. URL http://www.w3.org/TR/rdf-sparql-query/, http://www.w3.org/TR/rdf-sparql-query/.

Leslie N. Smith. 2015. No more pesky learning rate guessing games. *CoRR*, abs/1506.01186. URL http://arxiv.org/abs/1506.01186.

Wietse de Vries, Andreas van Cranenburgh, Arianna Bisazza, Tommaso Caselli, Gertjan van Noord, and Malvina Nissim. 2019. BERTje: A Dutch BERT Model. *arXiv:1912.09582 [cs]*. URL http://arxiv.org/abs/1912.09582.

Inc Wikimedia Foundation. 2019. Category:main topic classifications. https://en.wikipedia.org/wiki/Category:Main_topic_classifications. Verkregen op: 2020-08-01.

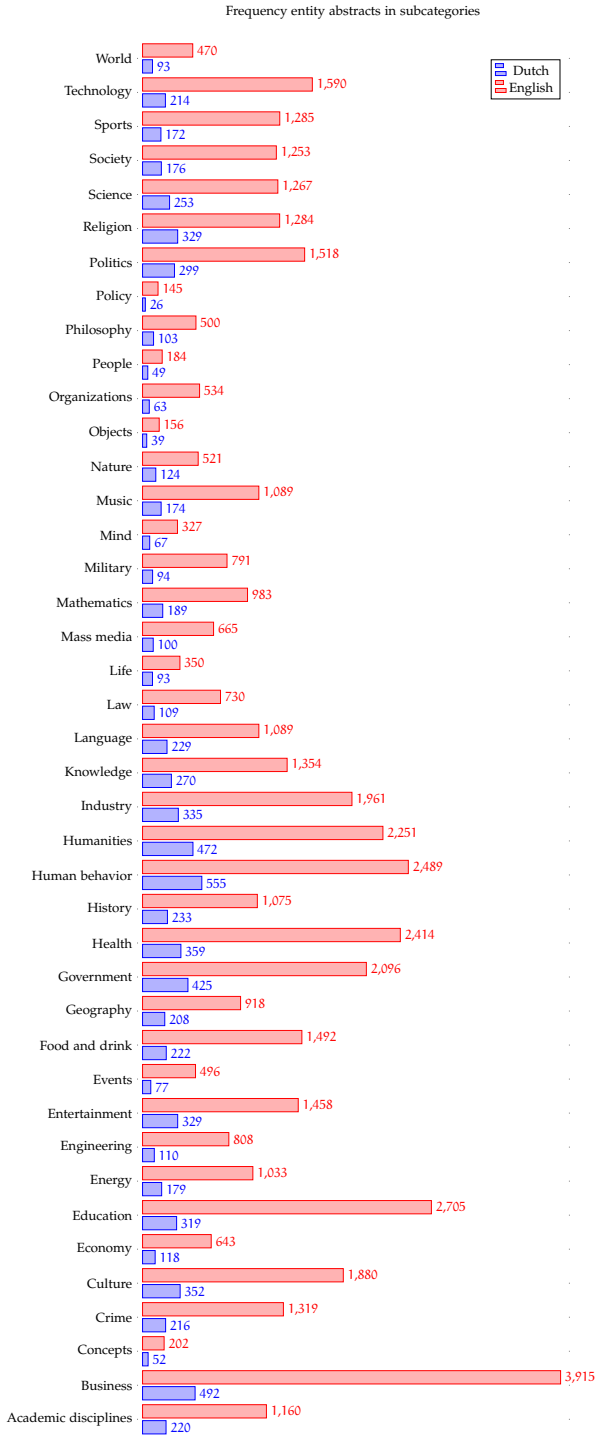
Congying Xia, Chenwei Zhang, Tao Yang, Yaliang Li, Nan Du, Xian Wu, Wei Fan, Fenglong Ma, and Philip Yu. 2019. Multi-grained named entity recognition. *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. URL http://dx.doi.org/10.18653/v1/P19-1138.

Shimin Zhong, Yajun Du, and Zhen Wei Gao. 2018. Fine-grained named entity recognition in question answering with DBpedia. *Journal of Physics: Conference Series*, 1087:032003. URL https://doi.org/10.1088%2F1742-6596%2F1087%2F3%2F032003.

Appendices

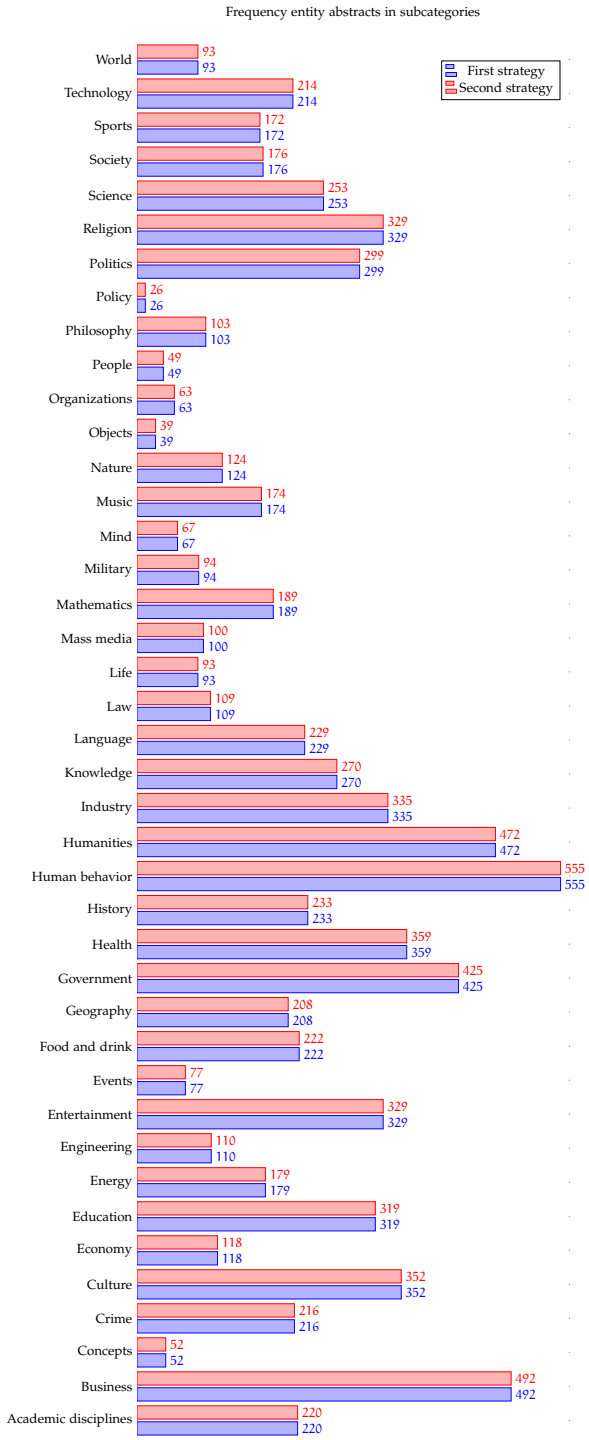
A APPENDIX

A.1 OVERVIEW DISTRIBUTION DUTCH VS ENGLISH



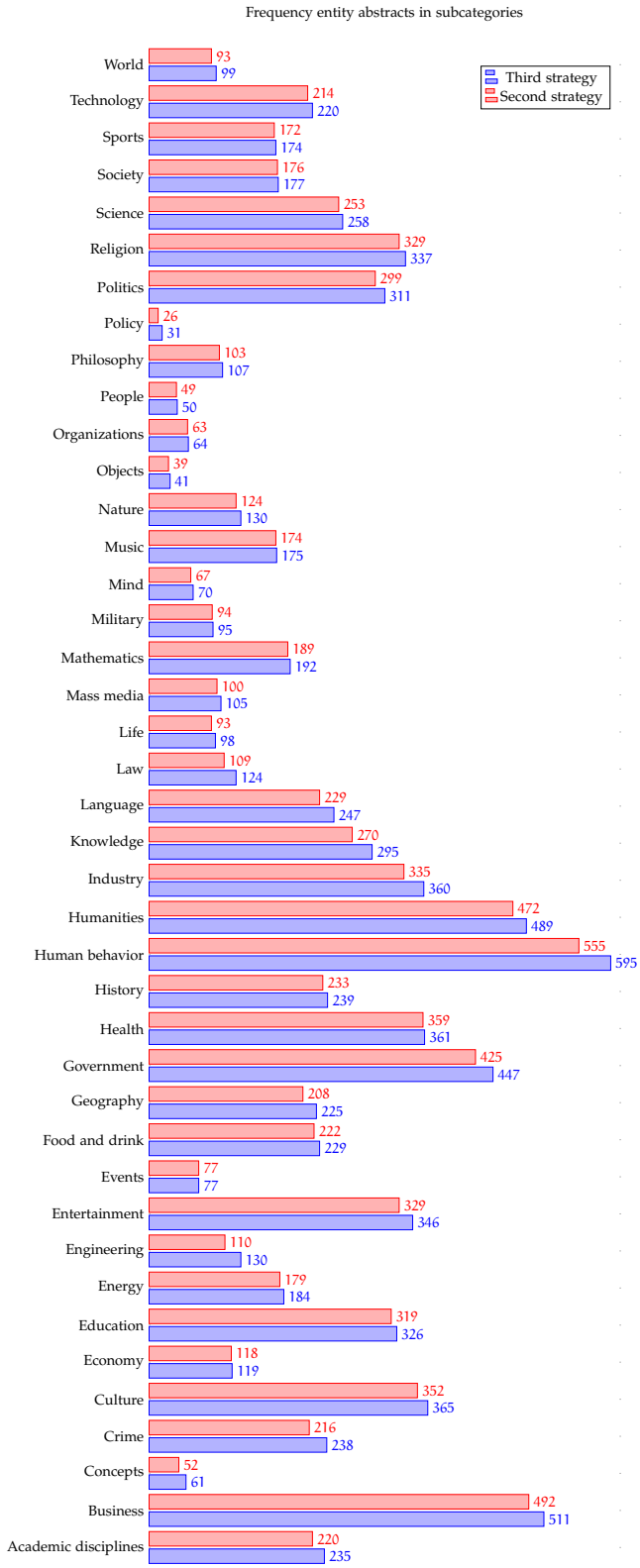
Graph A.1 This distribution table gives off the annotation labels and the frequency of the entities per MTC subcategory, excluding the MTC without subcategories for the Dutch language.

A.2 FIRST VS SECOND COLLECTION STRATEGY



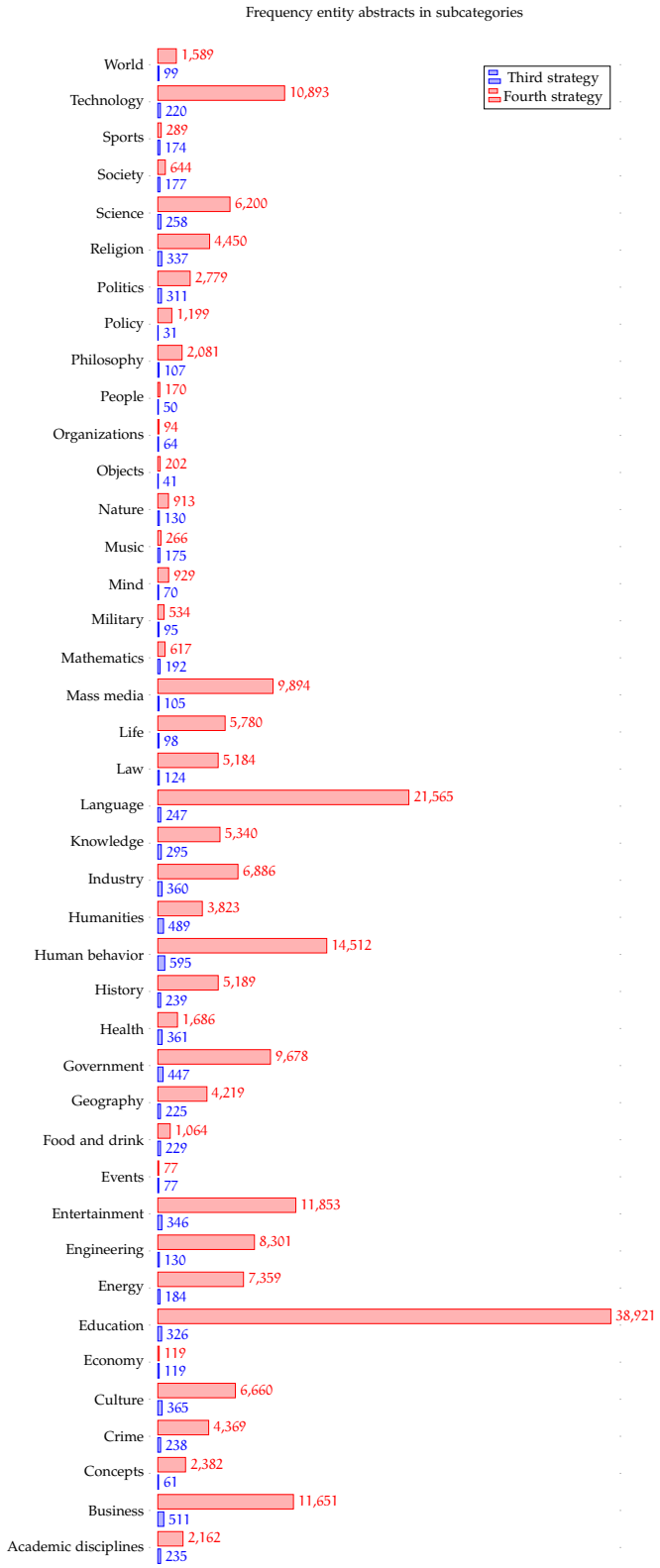
Graph A.2 This distribution table gives off the annotation labels and the frequency of the entities per MTC subcategory, excluding the MTC without subcategories for the Dutch language comparing the first strategy against the second strategy.

A.3 SECOND VS THIRD COLLECTION STRATEGY



Graph A.3 This distribution table gives off the annotation labels and the frequency of the entities per MTC subcategory, excluding the MTC without subcategories for the Dutch language comparing the second strategy against the third strategy.

A.4 THIRD VS FOURTH COLLECTION STRATEGY



Graph A.7 This distribution table gives off the annotation labels and the frequency of the entities per MTC subcategory, excluding the MTC without subcategories for the Dutch language comparing the third strategy against the fourth strategy.

A.5 DUTCH ENTITIES

Main Topic Classifications	Frequency entities in subcategories
Academic disciplines	220
Business	492
Concepts	52
Crime	216
Culture	352
Economy	118
Education	319
Energy	179
Engineering	110
Entertainment	329
Events	77
Food and drink	222
Geography	208
Government	425
Health	359
History	233
Human behavior	555
Humanities	472
Industry	335
Knowledge	270
Language	229
Law	109
Life	93
Mass media	100
Mathematics	189
Military	94
Mind	67
Music	174
Nature	124
Objects	39
Organizations	63
People	49
Philosophy	103
Policy	26
Politics	299
Religion	329
Science	253
Society	176
Sports	172
Technology	214
World	93
Total	**8.538**

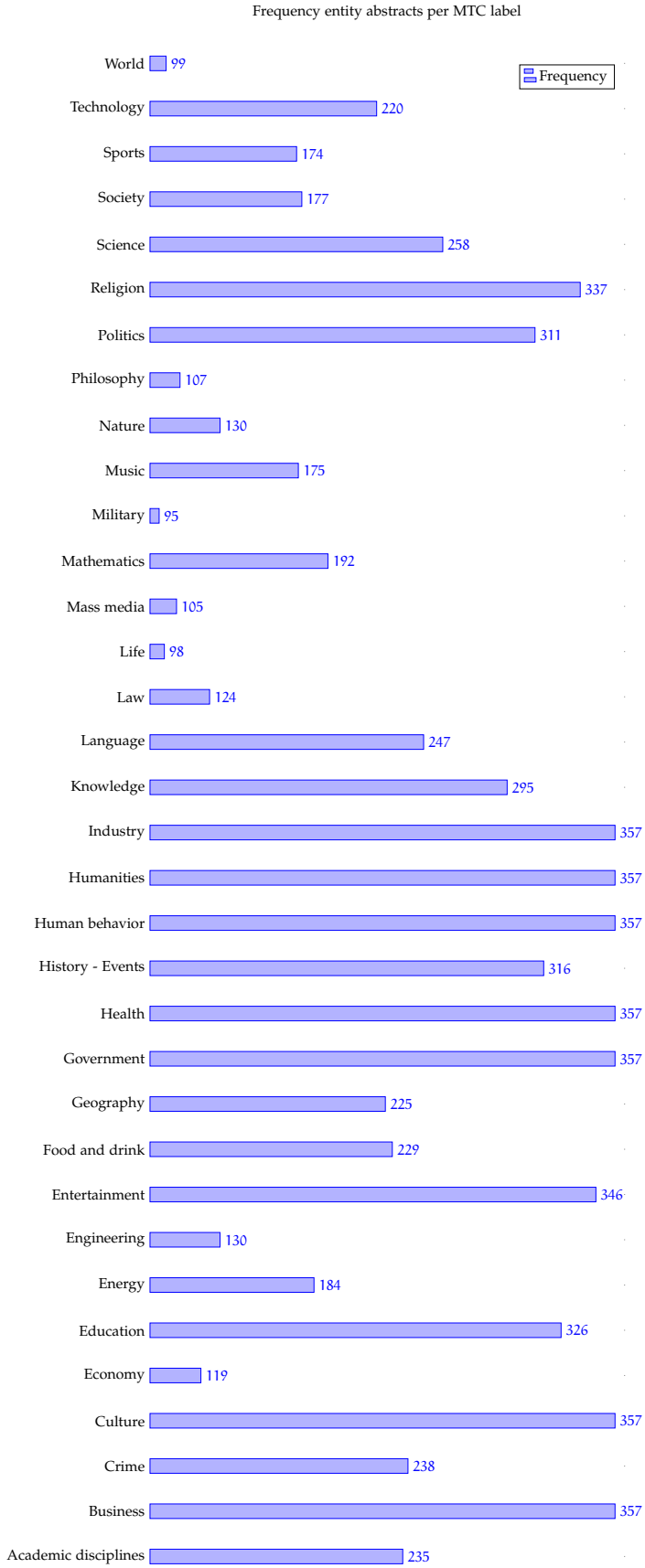
Table A.5 This distribution table gives of the annotation lables and the frequency of the entities that holds a Dutch abstract per MTC subcategory excluding the MTC without subcategories using strategy one.

A.6 ENGLISH ENTITIES

Main Topic Classifications	Frequency entities in subcategories
Academic disciplines	1160
Business	3915
Concepts	202
Crime	1319
Culture	1880
Economy	643
Education	2705
Energy	1033
Engineering	808
Entertainment	1458
Events	496
Food and drink	1492
Geography	918
Government	2096
Health	2414
History	1075
Human behavior	2489
Humanities	2251
Industry	1961
Knowledge	1354
Language	1089
Law	730
Life	350
Mass media	665
Mathematics	983
Military	791
Mind	327
Music	1089
Nature	521
Objects	156
Organizations	534
People	184
Philosophy	500
Policy	145
Politics	1518
Religion	1284
Science	1267
Society	1253
Sports	1285
Technology	1590
World	470
Total	**48.400**

Table A.6 This distribution table gives of the annotation lables and the frequency of the entities that holds a English abstract per MTC subcategory excluding the MTC without subcategories using strategy one.

A.7 MAIN TOPIC CLASSIFICATIONS FOR LABEL ALLOCATION



MTC	Total
1. Academic disciplines	235
2. Crime	238
3. Economy	119
4. Education	326
5. Energy	184
6. Engineering	130
7. Entertainment	346
8. Food and drink	229
9. Geography	225
10. History	239
11. Knowledge	270
12. Language	247
13 Law	124
14. Life	98
15. Mass media	105
16. Mathemetics	192
17. Military	95
18. Music	175
19. Nature	130
20. Philosophy	107
21. Politics	311
22. Religion	337
23. Science	258
24. Society	177
25. Sports	174
26. Technology	220
27. World	99
Total MTCs	5390

Table 9: MTCs that fall within the range of 70 - 357 entities.

MTC	Total before	Total lost
1. Business	511	154
2. Culture	365	8
3. Government	447	90
4. Health	361	4
5. Human behavior	595	238
6. Humanities	489	132
7. Industry	360	3
Total pruned MTCs	3128	629

Table 10: MTCs that are pruned to 357 entities.

MTC	Total entities	Total needed
1. Concepts	61	18
2. Events	77	2
3. Mind	70	9
4. Objects	41	38
5. Organizations	64	15
6. People	50	29
7. Policy	31	48
Total potentially removed MTCs	394	159

Table 11: MTCs that fall under the range of 70 - 357 entities.

Total entities	Total MTCs	Within range	Removed	Merged	Pruned
7.991	34	5390	946	77	629

Table 12: Overall status changes to dataset.