TALLINN UNIVERSITY OF TECHNOLOGY
School of Information Technologies
Department of Software Science

Siim Kaspar Uustalu 176647IAPM

AUTOMATED DETECTION AND SENTIMENT ANALYSIS OF REGISTERED ENTITY MENTIONS IN ESTONIAN LANGUAGE NEWS MEDIA

Master's thesis

Supervisors: Tanel Alumäe
PhD
Karl Märka
MSc

Tallinn 2019

TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond
Tarkvarateaduse instituut

Siim Kaspar Uustalu 176647IAPM

ÄRIREGISTRISSE KANTUD ASUTUSTE MAINIMISTE AUTOMEERITUD TUVASTAMINE JA MEELSUSE HINDAMINE EESTIKEELSES MEEDIAS

Magistritöö

Juhendajad: Tanel Alumäe
Doktorikraad
Karl Märka
Magistrikraad

Tallinn 2019

Author's declaration of originality

I hereby certify that I am the sole author of this thesis. All the used materials, references to the literature and the work of others have been referred to. This thesis has not been presented for examination anywhere else.

Author: Siim Kaspar Uustalu

May 6, 2019

Abstract

The main goal of this thesis is to design, prototype and evaluate an automated Natural Language Processing (NLP) system capable of performing automated detection and sentiment analysis on mentions of entities registered in the Estonian Business Register in Estonian language textual news media. The main task is divided into two sub tasks corresponding to well known problems in Natural Language Processing (NLP). The first sub task is entity detection based on Named Entity Recognition (NER) capabilities of the estnltk NLP toolkit [1] for Estonian language augmented with a Named Entity Linking (NEL) system backed by the Estonian Business Register. The second task is Entity-Level Sentiment Analysis (ELSA) on articles making use of the entity detection results, which is approached in the form of a document classification problem with some preprocessing. Due to the varying sizes of the automatically generated documents, pretrained word embeddings are experimented with to test whether their use during vectorization improves accuracy in automated sentiment analysis.

The developed system's design is approached in a general way. Division into smaller subsystems and later composition aids in both enabling isolation during the evaluation phase and potential repurposing for later processing tasks. Both developed artefacts are evaluated on a proprietary *ad hoc* corpus of 800 articles divided into a visible and blind partition, with only the larger visible partition made available to the author. Evaluations on the blind partition were performed by a separate evaluator. Achieved results are reported for experiments on both partitions of the corpus. Potential future improvements upon the current implementations are provided for both artefacts.

The thesis is in English and contains 66 pages of text, 5 chapters, 9 figures, 5 tables.

Annotatsioon

Äriregistrisse kantud asutuste mainimiste automeeritud tuvastamine ja meelsuse hindamine eestikeelses meedias

Käesoleva magistritöö eesmärgiks on luua automaatne loomuliku keele töötluse süsteem, mis tuvastaks Eesti Äriregistrist leitavate ettevõtete ja asutuste mainimisi eestikeelsetes meediaväljaannetes. Süsteem peab tuvastama olemid ja määrama tuvastatud olemite alusel ka olemite suunas väljendatava meelsuse. Töö probleem on jaotatud selguse ja üldistatavuse eesmärgil täiendavalt kaheks alamprobleemiks, mis kumbki keskendub aktuaalsetele loomuliku keele töötluse probleemidele. Töö esimene alaprobleem on mainitud olemite tuvastamine, mida käsitletakse nimega üksuste määratlemise ülesandena. Nimega üksuste määratleja on loodud täiendama olemasolevat nimega üksuste markeerijat. Töö teine alaprobleem on meelsuse hindamine, mida käsitletakse kui täiendavat eeltöötlust vajavat teksti klassifitseerimise ülesannet. Dokumendi tasemel meelsuse hindamine ei paku piisavat selgust ja tulemuste detailsust. Seetõttu on tarvis sooritada olemi tasemel meelsuse analüüsi, mis genereeriks eeltöötluse käigus iga olemi suhtes meelsust väljendava dokumendi.

Süsteemi disaini juures on silmas peetud üldistatavust. Tervikliku süsteemi jaotamine andmekonveierina kombineeritud kaheks alamülesannet realiseerivaks süsteemiks võimaldab hilisemat komponentide taaskasutust. Magistritöö käigus hinnatakse mõlemaid looduid artefakte 800 artiklilisel annoteeritud andmekogumil. Andmekogum on täiendavalt jaotatud nähtavaks ning varjatud jaotiseks. Arenduse käigus on autorile tehtud saadavaks ainult nähtav osa andmekogust ning varjatud jaotisel toimub hindamine täiendava hindaja poolt. Töö autor esitab artefaktide kohta saavutatud täpsuse meetrikad ja pakub võimalikke täiendusi juba seniloodud lahendustele.

Lõputöö on kirjutatud inglise keeles ning sisaldab teksti 66 leheküljel, 5 peatükki, 9 joonist, 5 tabelit.

List of abbreviations and terms

ALSA Aspect-Level Sentiment Analysis.

ANN Artificial Neural Network.

API Application Programming Interface.

AWS Amazon Web Services.

CSV comma-separated values.

ELSA Entity-Level Sentiment Analysis.

FaaS Function as a Service.

HTTP Hypertext Transfer Protocol.

JSON JavaScript Object Notation.

NEL Named Entity Linking.

NER Named Entity Recognition.

NLP Natural Language Processing.

POS Parts Of Speech.

SVM Support Vector Machine.

TSV tab-separated values.

Table of contents

List of figures

List of tables

1. Introduction

The thesis describes a practical exercise in design science research [2] to design, prototype and evaluate an automated Natural Language Processing (NLP) system capable of performing entity detection with linking and sentiment analysis on textual news media mentions of entities registered in the Estonian Business Registry. The work has been divided into two subtasks for the purposes of the thesis. First, the mention discovery which consists of entity detection along with linking the detection results to specific entries in the knowledge base. The second task is classification of the sentiment carried by those mentions into one of three possible sentiment classes: $negative$, $neutral$ and $positive$. Software artefacts are developed and then subsequently evaluated on a proprietary article corpus of around 800 articles provided by the commissioning party.

1.1 Motivation and problem background

The thesis topic was proposed by Creditinfo Eesti AS, which is a credit bureau offering credit scores for private persons and companies along with general market analytics. Their aim is to provide added value on top of aggregated information to enable clients to make informed and timely financial decisions, control potential debt and increase revenue [3]. Their software suite provides a comprehensive report on the financial performance of registered entities enabling partner entities and financial institutions to evaluate their solvency. These reports aggregate information from a number of sources such as the Estonian Business Register, news media and Official Announcements [4], [5].

Some parts of information gathering have already been automated while others, such as annotation of company profiles with mentions in media sources, involve a notable amount of manual work. Currently, human operators manually review published news articles using specialised tools to annotate mentions of corporate entities, map the mentions to corporate entity profiles and update the relevant database with a paragraph from the article where the mention occurred. The sentiment of mentions is not evaluated as part of the manual process. Sentiment analysis provides added value as a standalone media monitoring product for marketing and public relations teams [6]. Further, sentiment towards an entity may provide an additional signal in the credit scoring process [7].

1.2 Research goal

The final goal of the work done in the creation a new software system enabling automated processing of news media to discover and annotate mentions of entities registered within the Estonian Business Register. The larger software system artefact consists of multiple artefacts composed as a processing pipeline. Each component is responsible for a single sub problem described in this chapter. Division into separate artefacts enables potential re-use of components and isolation during the evaluation process.

1.2.1 Research questions

- RQ - How to create a natural language processing pipeline from the artefacts developed to solve subproblems as defined at the start of chapter 1?
- RQ 1 - How to augment a general named entity recogniser to improve detection accuracy?
  - RQ 1.1 - What other language processing signals can be used to discover entities missed by the named entity recognition module?
  - RQ 1.2 - What possibilities are there to combine named entity recognition with named entity linking in case of a closed set of possible entities?

- RQ 2 - How to link detected entity candidates to entries in the authoritative knowledge base?
  - RQ 2.1 - What method can be used to normalise corporate names to enable look up from the registered entity database?
  - RQ 2.2 - What data structures enable efficient text based lookup of registered corporate entities?
- RQ 3 - How applicable is English language based research on Entity-Level Sentiment Analysis within the context of Estonian natural language processing?
  - RQ 3.1 - What approach seems most applicable to the problem tackled within the scope of this thesis?
  - RQ 3.2 - What accuracy is achievable on the provided hand annotated evaluation data set with the chosen approach?

2. Theoretical background and related work

In this chapter, the theoretical background for the problems introduced in chapter 1 is explored. Further divisions are applied to the problems in order to enable direct relation to well known problems in NLP. The first section focuses on the problem of entity detection and how it can be further divided into two well known problems in NLP. The second section introduces the sentiment analysis problem along with its entity level variation. Finally, a vector space representation of words encapsulating semantic information is introduced as part of featurization theory for sentiment analysis.

2.1 Named entity recognition and linking

The detection of registered entity mentions encompasses two well known problems in the domain of natural language processing. First of them — detection of potential entity candidates is similar to the well known problem of Named Entity Recognition (NER), which was originally formulated in the 1990s as part of the Message Understanding Conferences [8], [9]. NER can be defined as a sequence tagging problem, where for a given sequence of text, its tokens are tagged as belonging to one of the possible entity categories (such as people, locations or organisations) or to none [10]. A visual example of NER is provided in Figure 1. Software packages implementing NER among other NLP tasks are available for a multitude of human languages. Often the software packages provide support for multilingual NER by supporting multiple language models [11], [12]. A widely used open source NLP toolkit, called *estnltk*, exists for the Estonian language. Similarly to other NLP toolkits, it includes a module providing NER capabilities [1]. A limiting factor of it is that only three entity categories are supported: *person (PER)*, *location (LOC)* and *organisation (ORG)* [13].

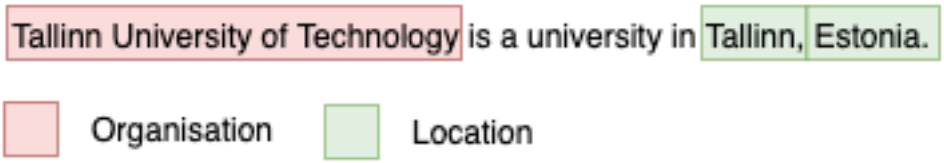

Figure 1: An example of Named Entity Recognition in action

The NER implementation in *estnltk* uses a supervised machine learning approach with the conditional random fields model. It must be noted that no reference corpus for Estonian language NER existed prior to the development of this module. A new reference corpus was compiled and made available by the authors [13], [14]. The NER module is reported as achieving a F_1-score of 87.0% over all entity categories, but only 77.1% for the organisations (*ORG*) category of named entities [13]. The per class precision and recall for *ORG* entities were reported as 80% and 74.7%, respectively. Direct comparison to NER model performance on other human languages is not possible due to linguistic differences. Complexity of the language, a lesser variety in corpora and availability of linguistic tooling— these are all factors contributing to inferior results achieved as compared to more well-studied languages such as English [14].

Loose comparison can be made to the performance on similar tasks in other human languages. In order to establish a point of comparison, for the common English language NER task (CoNLL 2003 [15]), the state of the art overall F_1-score, at the time of writing, was reported as 93.5% [16] achieved with an Artificial Neural Network (ANN) based deep learning approach. Notably, for Hungarian, another member of the Uralic languages family, the state of the art NER F_1-score is reported as 95.06% achieved on the business news domain Szeged NER corpus [17] using a maximum entropy classifier [18]. It must also be noted that the scores are largely affected by the corpus on which the model is trained and applied to. As a result of the relatively low accuracy metrics for the Estonian language, as cited previously, the entity discovery process proposed in this thesis can not only rely on the *estnltk* NER module — a specialised sequence tagger or some form of augmentation on top of existing tooling is needed.

Training a specialist model for NER is possible by making use of existing *estnltk* features [19]. It must be noted, that the original Estonian NER model was trained on a corpus of 572 news stories [13]. Experiments in this thesis are performed on a similarly sized dataset of 560 articles retrieved from online media, but no token level annotations are available. Further, given the thesis problem formulation, even the most similar NER category, *ORG*, does not directly map to the class of entities that are of relevance as only

entities that have been registered within the Estonian Business Register are of interest. Taking all this to account means that improvement over the state of the art is unlikely by merely training a new model given the limitations.

The augmentation approach leads to another well known problem in NLP — Named Entity Linking (NEL). Named Entity Linking refers to the process of matching discovered named entity mentions to entries in a given named entity knowledge base [20]. Within the problem context of this thesis, the authoritative entity database is a snapshot of the Estonian Business Register. Currently, the snapshot is restricted to only corporate entities active at the time of snapshot creation and does not contain previously used entity names or historic entities. Use of this database enables the implementation of the dictionary based approach to Named Entity Linking (NEL), where the knowledge base is a list of $< entity\ name, entity\ identifier >$ mappings. Fuzzy matching is then performed on the entity names for each entity candidate to determine whether a related entity exists and the discovered entity is mapped to the appropriate knowledge base entry [20]. The relatively small size (282 407 entries as of 01.01.2019 [21]) of the knowledge base enables effective string based in-memory lookup with the use of a specialised data structure such as a trie [22] or its suffix tree [23] variant. NEL can then be used to augment the NER process and restrict the output to only those entities that can be mapped to knowledge base entries.

2.2 Sentiment analysis

Sentiment analysis is a text classification problem, where NLP techniques are used to determine the opinions (or sentiment) expressed by the authors of a text passage towards the subject of the text [6]. Another task in sentiment analysis is the classification of the general emotions conveyed by a text passage [6]. It has applications in domains where the sentiment expressed by external parties matters. For example: social media analysis, advertising, governmental intelligence [6] and notably, credit scoring [7]. Tasks in sentiment analysis range from binary labelling problems, where sentiment polarity is expressed as two opposites: *negative* and *positive* to more complex tasks providing a more granular sentiment assessment [6]. Instead of labelling text with discrete values, the problem could also be approached as a regression task where a numeric sentiment score is provided on a scale between the two opposites [6]. A more complex and recent task is to indicate the emotional state of the writer based on the basic human emotions classifier (for example Plutchik's wheel of emotions [24]) [25]. This form of task is most relevant in social media based opinion mining, as the communication styles present in social networks are less

formal as compared to traditional media sources [26]. Beyond classification, the emotional valence (intensity) of expressed sentiment can also be predicted based on detection results [6], [25]. With regard to techniques, both naïve dictionary based approaches are in use along with both unsupervised and supervised machine learning techniques [6].

Another factor that must be taken into account in sentiment analysis and opinion mining is the subjectivity of sentiment expression [27]. It can not be objectively observed nor verified and depends on the author [6], [27]. A demonstration of this property is the differing opinions of coders during the annotation phase [28], [29]. Beyond subjectivity, sentiment varies based upon the level of observation. Sentiment can be observed on the larger body of text as a whole (document-level) or in a more granular fashion on its sub-document units (sentences and paragraphs) as sentiment of each may differ. The document-level sentiment label can then be thought of as a function of the sentence and paragraph level labels [6]. Another approach to granularity is to determine entities and/or aspects of entities (such as the expected battery life of a mobile phone) and then label the sentiment towards each. The described approaches to sentiment analysis are referred to as Entity-Level Sentiment Analysis (ELSA) and Aspect-Level Sentiment Analysis (ALSA) respectively [6].

In the problem context of this thesis document-level sentiment analysis does not provide the necessary level of information granularity. An article may contain mentions of multiple entities with varying sentiment polarity towards each. Given that the NER and NEL portions provide a set of discovered entities makes it possible to perform ELSA. Most focus in ELSA research has understandably been towards analysis of opinionated texts, such as reviews in various domains [30]–[32] and microblogs [33]. In the news media context, sentiment analysis has mostly been tried by applying either a sentiment lexicon based (one such lexical resource is sentiwordnet [34]) approach to score general sentiment expressed by articles [35] or determine sentiment towards each mentioned entity [36]. In the lexicon based approach the immediate surroundings of an entity mention are gathered and a dictionary lookup is performed to evaluate the sentiment for each. The final sentiment assessment is then determined by combining the word-level results. A similar approach makes use of specially constructed role specific dictionaries and instead of determining a sentiment score or label, tries to identify the entities that are conveyed as belonging to one of the three roles prevalent in news media (hero, victim and villain) [37].

2.2.1 Estonian language sentiment analysis

In Estonian language sentiment analysis research, both lexicon based and machine learning approaches have been experimented with [38], [39]. In comparison to English language research, the focus has not been on reviews, but more towards determination of sentiment expressed in news media [38] or in conversational texts [39]. Although Pajupuu *et al.* also performed sentiment analysis on a text corpus compiled from online news media and comments, only paragraph-level sentiment annotations were prepared and ELSA was not attempted [38]. This means that ELSA has a degree of novelty when applied to Estonian language news media. To provide a point of reference of the accuracy achieved by earlier work, the results reported by Pajupuu *et al.* of $75.5\% \pm 10.8$ overall accuracy (using a linear kernel Support Vector Machine (SVM) classifier with term frequency - inverse document frequency (TF-IDF) vectorization) [38] are considered state of the art for the purposes of the thesis. Even though Ojamaa *et al.* report *an overall correctness* of 85% [39], the conversational texts domain enables better expression of sentiment and as such is not comparable to the problem at hand.

In regard to the availability of sentiment analysis corpora, there exists the Estonian valence corpus [40] compiled as part of [38], that contains paragraph-level sentiment annotations on texts originating from online media sources. A variant of the Basic Estonian Dictionary where word-level sentiment labels from [38] have been applied to the around 3000 most common Estonian language words is also available as a lexical resource to facilitate lexicon based sentiment analysis approaches. A more recent effort focused on combining Estonian wordnet synsets with sentiment information from the English language *sentiwordnet* to create a new lexical resource, but did not produce promising results in its pilot study [41].

2.2.2 Featurization for ELSA

In terms of featurization, compared to document-level sentiment analysis approaches, ELSA requires an additional preprocessing step where the document is processed to produce a set of features for all detected entities. An example of such an approach is *EntityWords*, where unigrams appearing in the same sentences as entities are [42] gathered to form sub-documents (referred to *context documents* in the scope of this thesis). The constructed context documents are of varying size ranging from a few unigrams up to hundreds, dependent on article length and the number of times the entity was mentioned.

Multiple methodologies exist to convert documents into feature vectors for use by machine learning models. If all text documents were of the same length, then a trivial example would be to construct a vocabulary mapping n-grams from the sequence to numeric values and construct vectors by replacing occurrences with their numeric representations. Documents rarely are of the same length and as such the trivial example methodology has low practical value. More commonly, feature vectors are constructed so that each dimension maps to a n-gram appearing in a vocabulary. The dimensions could then contain appearance counts, frequency or simply presence indicators [6]. A popular vectorization algorithm is TF-IDF, as it gives more weight to words that are more common in a single document or small group of documents but infrequent in the whole corpus [43].

The previously described methodologies result in rather high dimensional vector representations of documents or are not practically usable. With the more prevalent usage of ANN for NLP tasks, using ANN to create vector space representations of words and documents is becoming more common. Specialised ANN are trained on large collections of data to produce comparatively low dimensional continuous vector space representations of words. There are two notable properties of these vector representations. The first being that similar words are positioned nearby in the vector space and the second, that algebraic operations on the representations take on semantic meaning [44]. An example of semantic meaning being conveyed through algebraic expressions is given in Figure 2. These properties lead to a hypothesis that use of word vectors may help derive more semantic information from context documents shorter than a paragraph. For the Estonian language, *word2vec* word embeddings pretrained on the Estonian Reference Corpus [45] have been made available. Other multilingual language models that produce vector representations, such as *BERT* [46] are also available in a pretrained form. An additional bonus of word embedding usage is that it makes the classifier more flexible and not only depend on the vocabulary of the training corpus. The classifier will learn from the traits exhibited by the word embeddings and can then even perform classification on documents that contain no previously seen n-grams.

$$vector(\text{"}king\text{"}) - vector(\text{"}man\text{"}) + vector(\text{"}woman\text{"}) \approx vector(\text{"}queen\text{"})$$

Figure 2: An example of algebraic operations on word vectors

3. Methodology

In this chapter the methodologies used to approach the previously described problems are described. The first section focuses on the general architectural principles that affected the design of the media monitoring system. The second section describes the proprietary *ad hoc* article corpus used for evaluation purposes in this thesis. The final two sections describe the entity detection and sentiment analysis services. A detailed description of the methodology used to discover registered corporate entity candidates and map them to entities appearing in the Estonian Business Register is given in the third section. The final section focuses on the problem of ELSA in the domain of Estonian language online business news.

3.1 Design of the media monitoring system

The subject of the thesis is an applied design research project to implement two processing components of an automated media monitoring system. The larger task to be achieved by the system can be worded as follows: given the text of an news article, find mentions of valid corporate entities and determine whether the sentiment expressed in the article towards each entity is $negative$, $neutral$ or $positive$. Division of the larger media monitoring task into subtasks corresponding to well known NLP problems was previously discussed within chapters 1 and 2. This natural separation then enables creation of focused subsystems in line with the principles of the service oriented architecture approach to software development. These separate services are then composed as a pipeline to achieve the main research question (RQ) proposed in section 1.2. A birds eye view of the media monitoring system's architecture is given in Figure 3.

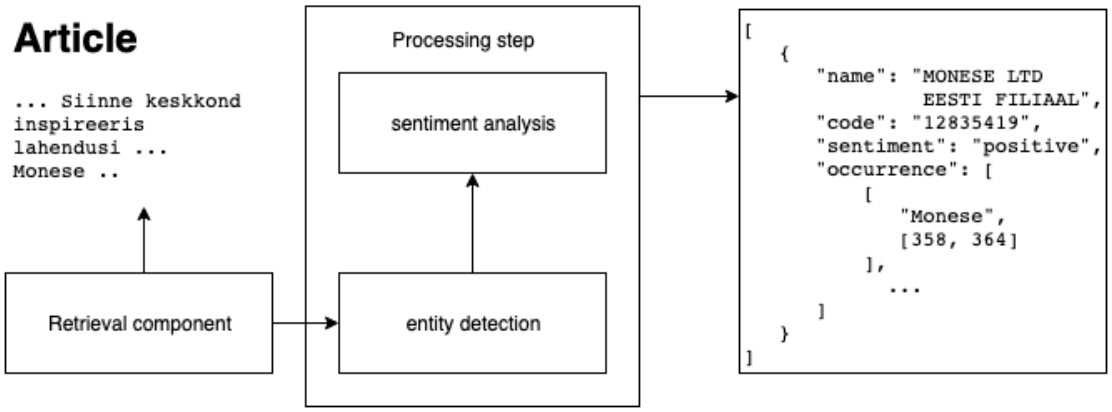

Figure 3: High level architectural overview

There are two notable design considerations contributing to the choice of the pipeline based approach. First of them is the ability to perform performance evaluation of system components in isolation. The second consideration is the ability to repurpose the specialist services for other tasks. Adoption of a service oriented approach means that the entity detection component presented in Figure 3 could at a later date be replaced by a service implementing a different strategy to monitor for other classes of occurrences while still making use of other pipeline components. Entity detection results have a wider range of potential application than the current downstream problem of ELSA.

3.1.1 Technical implementation considerations

In addition to the architectural design considerations stemming from functional requirements and the potential opportunities to repurpose as part of potential future work, another set of design considerations influencing the technical choices for implementation exists. One of the largest limitations stems from the fact that *estnltk* is distributed as a Python library [1]. A wrapper could be developed around the library to make it callable from code written in other languages, but the required effort is not warranted at this time. Python itself is a rather widely used language in the data science domain. Additionally it is used widely enough that all software libraries needed for service oriented development are available.

The chosen service oriented approach to the problem, as can be seen in Figure 3, necessitated the choice of a communication protocol between the services. Instead of a machine readable protocol or integration with an external message queue, the widely used and human readable JavaScript Object Notation (JSON) over Hypertext Transfer Protocol (HTTP) approach was chosen due to its ease of implementation. Further, the predicted usage of the pipeline services did not warrant the choice of a more complex and performance oriented inter service communication protocol.

The deployment environment was also given early consideration and influenced some design decisions. The Creditinfo team makes use of the Amazon Web Services (AWS) cloud platorm and as such the AWS Lambda service was originally considered as the target deployment environment. AWS Lambda is a Function as a Service (FaaS) solution where the underlying infrastructure (server and operating system) have been abstracted away [47]. The platform was considered due to AWS Lambda's potential to lower the operational costs of the media monitoring system due to subsecond compute resource usage metering fitting the operational characteristics of a data processing pipeline[47].

After further consideration, the AWS Lambda FaaS platform proved unsuitable due to certain *estnltk* components requiring native compilation and runtime requirements not fulfilled by AWS Lambda containers designed for Python deployments. Further, certain *estnltk* features required the Java runtime environment to be present in addition to the Python runtime. Instead, a containerisation based approach was chosen as the delivery format of the resulting system artefacts composed of code and required resources. Build processes, that package the created software into Docker images for distribution purposes were developed for both subsystems. These images can then be utilised on various target platforms to provision system components [48].

3.2 The proprietary article corpus

Part of the project included the creation of an *an hoc* data set against which development and final evaluation of the system artefacts could be performed. Annotations were performed by a qualified employee of Creditinfo routinely performing similar tasks as to the ones attempted by the tool under development. In total the corpus consisted of 800 Estonian language online news articles. Early on the decision was made to divide the corpus into two sub sets: visible and blind. The visible set is larger and consists of 560 annotated articles compared to 240 articles in the blind portion.

The visible partition of the corpus was made available to the thesis author to perform experimentation and initial performance estimations during the development phase. The smaller partition, referred to as the blind set was only used for final performance evaluations on the developed artefacts. The results achieved on the blind set were not used to further develop the system as part of the work done in this thesis. Overall this sort of workflow is loosely similar to a Kaggle competition [49].

The corpus itself consists of a tab-separated values (TSV) format text file containing annotations where each line contains the annotations for a single article and links to a text file containing plain text article contents. A partial example of annotation entries is given below in Table 1 with some extraneous columns omitted for brevity. The *positive*, *negative* and *neutral* columns all contain comma separated lists of entities grouped by the sentiment towards them. It must be noted that only a single coder was used to determine ground truth sentiment annotations.

article	**positive**	**negative**	**neutral**
1.txt	MONESE		
118.txt	FANVESTORY		GEENIUS
163.txt	TAXIFY,TRANSFERWISE		ÄRIPÄEV

Table 1: Partial example of multiple annotation entries

The above example in Table 1 highlights an issue effecting the evaluation process within the thesis. The annotations were compiled by a qualified media monitoring worker without specialist oversight and as such the annotations are geared more towards human understanding. Usage of registration codes or fully qualified entity names would have reduced the amount of manual preprocessing required before automated artefact evaluation could be performed. Although this was an acceptable trade off as manual mapping of discovered entities to their registry codes is a time and attention intensive task. The required annotator effort would not make sense in the context of an ad hoc proprietary data set creation.

3.3 Entity detection methodology

The problem of entity detection was described in section 2.1 as a combination of the well known NER and NEL problems. In the scope of this thesis, focus is given only to entities registered in the Estonian Business Register. This means that the first step of the detection process is to generate token sequences that may refer to an entity. Given the generated token sequences, the second step is a mapping step where each generated sequence is attempted to be mapped to an entry in the source entity knowledge base. Once the mapping has been performed, different mentions of the same entity are aggregated under the detected normalised identifier.

The chosen approach sacrificed precision for recall as the business requirements were more permissive towards false positives over false negatives. Compared to development of a new custom NER model for the Estonian language, the chosen approach required less annotation effort — articles were only annotated as to the entities mentioned and sentiment polarity for each. It must be noted that the chosen approach does require the existence of multiple blacklists of common words and terms in order to reduce some of the more common false positives.

The resulting software artefacts from entity detection phase are a Python package providing entity detection capabilities and a HTTP Application Programming Interface (API) capable of accepting plain text articles for processing. The API returns a JSON dictionary of discovered entities and the textual mentions of these entities. An example of the API response is listed in appendix A. The output of the algorithm described within this section is used as part of the input for the sentiment analysis phase of processing.

3.3.1 The knowledge base

In NEL tasks authors often use entries from ontological databases such as Yago or Conceptnet to refer to entities. In other cases, Wikipedia may also be used as an authoritative entity database [20]. In the scope of this thesis the Estonian Business Register is used. It provides a mapping of registered business names adhering to a set of rules [50] to unique numeric identifiers (registry code). This means that it fulfils the necessary criteria required to be an entity knowledge base. In addition to registered corporate entities, the knowledge base contains information about political parties, governmental institutions, non governmental organisations, local branches of foreign corporations and sole proprietors [51].

The NEL implementation was developed to use a file based dump of the Estonian Business Register formatted as a comma-separated values (CSV) file. The file contents were then loaded into memory as a trie [22] based data structure using the implementation available from Google's *pygtrie* [52] library. It must be noted that sole proprietors were ignored as they are named after the proprietor increasing the potential for false detection due to the existence of namesakes among people (and a tendence of some people to exploit this relationship). To facilitate lookup, the entity names were case normalised to upper case and used as keys pointing to the numerical identifiers.

One limitation of the trie data structure is that only prefix search is possible by default. On occasion, certain entities may be referred to by the suffix of their registered business name. To give an example, *Advokaadibüroo NotAName* could be referred to as simply *NotAName* in articles by omitting the company type indicating prefix *Advokaadibüroo*. In order to handle this limitation a suffix tree [23] inspired addition enabling suffix based lookup was implemented on top of the trie data structure. Multi token entity names were split into individual tokens and entries were created for all suffix variations with minimal length of two tokens pointing towards the fully qualified entity name. A visual demonstration of the suffix tree based addition is provided in Figure 4. Each key now refers to a 2-tuple where the first value is either the numeric entity identifier or *None* and the second value is a list of keys the current key is a suffix of.

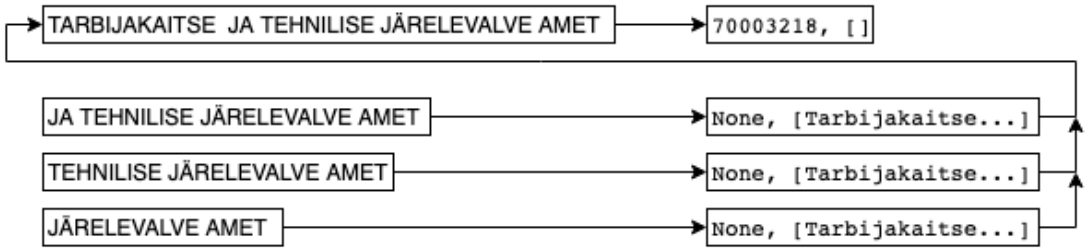

Figure 4: Example of the trie addition to enable suffix based entity lookup

3.3.2 Candidate generation

Initially the *estnltk* NER module's output for the *ORG* category was used to find token sequences that potentially refer to corporate entities registered in the Estonian Business Register. In early tests on collected articles it became apparent that this strategy would result in false negatives when solely relying on the output of the *estnltk* NER module. Although lacking support for the Estonian language, the *spaCy* named entity recognizer [12] was also trialled in combination with the *estnltk* NER module. Their combination was found to produce false positives causing confusion in the NEL process.

Instead, the author proposes a rule based approach for entity candidate generation that uses output of the NER module as one of multiple input signals. The input text is tokenized and tagged with syntax information using *estnltk*. NER is also performed on the processed text. Based on Parts Of Speech (POS) analysis an initial set of potentially interesting token sequences is generated. The subset of *estnltk* POS tags considered of interest is listed below in Table 2. Additionally, certain whitelisted tokens belonging to other POS categories are included.

POS tag	Description
A	adjective (positive form)
H	proper noun
N	cardinal numeral
O	ordinal numeral
S	noun
Y	abbreviation

Table 2: The subset of *estnltk* POS tags considered of interest

Subsequences of at least one token length are then generated from the token sequences in preparation for processing. The generated subsequences are sorted in descending order based on their length. If the first token in sequence is a legal form abbreviation (such as *OÜ* or *AS*), then another permutation where the abbreviation token is placed last is additionally generated This is done to match the normalised name form present in the CSV formatted dump of the Estonian Business Register.

3.3.3 Special considerations

Before moving on to the process of matching token sequences to entries in the entity knowledge base some special considerations need to be discussed. Although there are rules as to the naming of corporate entities, they are not that restrictive. This has lead to a situation where there are registered corporate entities with names that contain or consist of common words or phrases in Estonian and other languages. In order to reduce false positive matches resulting from these entities, a blacklist of commonly appearing tokens is required. This is achieved by populating an in memory trie with the 1000 most common Estonian words, 100 most commonly used Estonian language words in media texts [53] and the lemma list compiled by the Estonian Language Institute [54]. This lookup list is then used during the matching process to determine whether the mapped entity mention was deliberate. Common token sequences (sequences where each token is present in the blacklist) without an accompanying corporate form indicator are ignored due to their high false positive probability.

The two other special considerations are the cases where instead of legal corporate entity name, an *a priori* known to readers abbreviation or colloquial name is used. An abbreviation is usually used along with at least a single use of the fully qualified name, but this may not be the case with oft-used abbreviations. One example of such an abbreviation would be *EKRE* that is used in reference to *Eesti Konservatiivne Rahvaerakond*. The case with colloquial names is rather similar, but instead of an abbreviation, a shorter name that is commonly known to readers is used to refer to the entity. These names can sometimes even be a single token with no other usage. An example of this would be coverage of the Estonian national carrier *Nordica*, where the fully qualified business name (*Nordic Aviation Group AS*) is rarely used. The solutions to both of these special cases are similar. Dictionaries of well known and often used abbreviations and colloquial names are used. In the current implementation JSON encoded files are used to store both dictionaries. The contents are then loaded into memory to enable dictionary based lookup.

3.3.4 Linking to knowledge base entries

Once the potentially interesting token sequence permutations have been generated they are processed in order to determine whether they match any entity knowledge base entries. There are two general cases that require handling: multi-token sequence and single token. The exact handling of these cases varies, but shares some similarities.

In the multi-token case, the name string permutations are generated based on the present text and lemmas of all tokens included in the multi-token subsequences. If none of the tokens was tagged as a noun by the POS tagger, the subsequence is discarded. Another check determines whether any title cased tokens or legal form abbreviations are present. Subsequences without those tokens or consisting of only blacklisted tokens (see section 3.3.3) are also discarded. The generated strings are referred to as candidate strings and are used to query the entity knowledge base in a prefix based fashion to generate candidate entities. The candidate entities are then tested using regular expressions to determine whether they match the expected entity name forms based on the multi-token sequence.

Handling the single token case introduces another special case that is similar to the colloquial names special case outlined in section 3.3.3. Governmental agencies are often referred to by using colloquial names (such as *maksuamet* in reference to *Maksu- ja Tolliamet*) that can be used to match entities in a rule based fashion. When countering a compound word, the suffix is checked against a list of suffixes found in names of governmental agencies (such as *amet* or *inspektsioon*). If the suffix indicates a governmental agency, the prefix is used to query the in-memory register and a regular expression is used to validate that all parts of the compound word are part of the governmental entity's fully qualified legal name.

In the regular single token cases, the candidate strings are generated similarly to the multi-token case. A further dictionary based lookup is performed to determine whether the token is an abbreviation or a colloquial name. Before performing entity lookup, another check is done to determine whether the token under consideration is part of a multi-token sequence and such tokens are discarded as most likely they have already been processed. Once the checks have been performed, the final lookup and matching procedure is the same as in the multi-token sequence case. Candidate entities are returned from the entity knowledge base and the names are processed using regular expressions to determine whether they match the expected forms. The entity mentions are then gathered and aggregated under a single entity with the recorded occurrences. An example of the detection results returned by the entity detection API as a JSON encoded object is listed in appendix A. A holistic overview of the entity detection flow is presented in Figure 5.

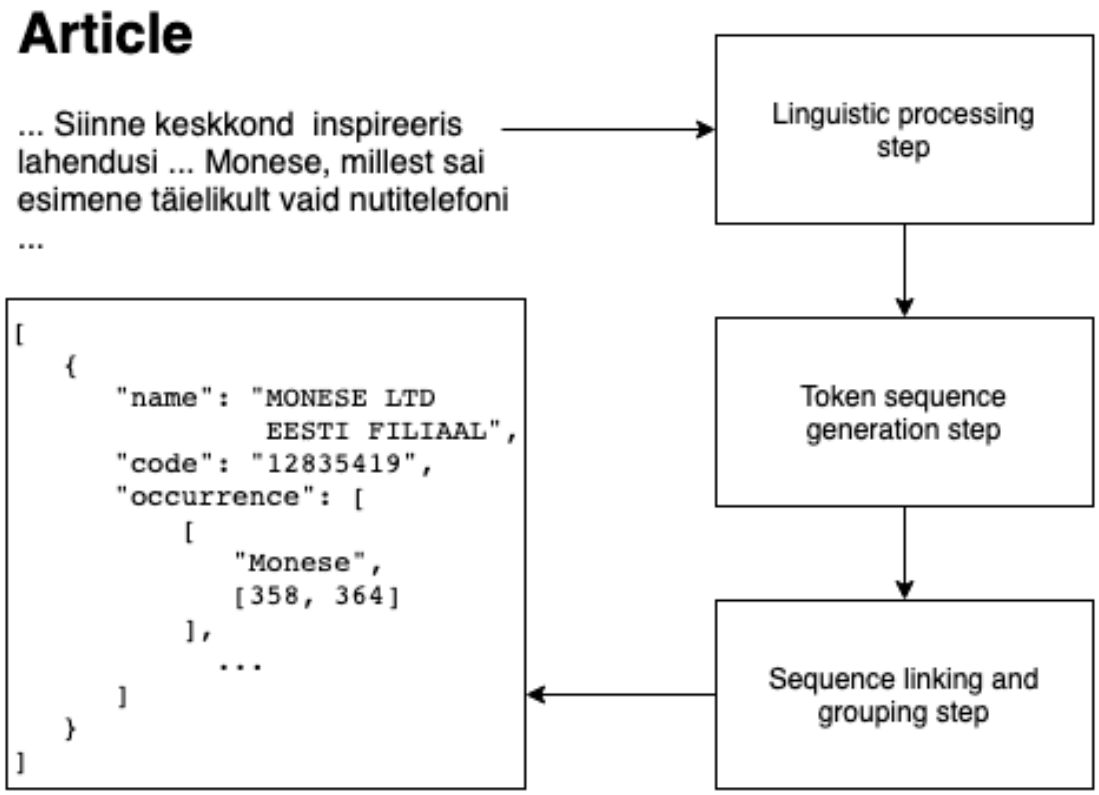

Figure 5: Overview of the entity detection flow

3.4 Sentiment analysis methodology

The sentiment analysis problem is approached as a three class text classification problem using supervised machine learning methodologies. Document-level sentiment analysis was found to not offer the necessary level of insight into sentiment within articles — polarity towards each mentioned entity may vary within a single document. This means that it is necessary to perform Entity-Level Sentiment Analysis. The problem was then formulated as follows: given a set of detected entity mentions and their appearance locations within the document, sentiment polarity towards each entity must be classified as $negative$, $neutral$ or $positive$.

3.4.1 Preprocessing for ELSA

When compared to regular document-level sentiment analysis, ELSA requires another preprocessing step before featurization can be performed. In the current methodology,

the problem is approached as a document classification problem over multiple generated documents. This necessitates the creation of documents corresponding to each entity that are representative of the sentiment conveyed in the article towards the entity. In the current implementation, a rather naïve approach called *EntityWords* [42] was used.

In the *EntityWords* approach, documents are created for each entity that consist of the tokens appearing in the same sentences as the detected entity mentions [42]. These documents are referred to from here on out as *context documents*. Multiple strategies were implemented to construct these context documents. These strategies are evaluated within chapter 4. A simplistic strategy gathers all of the tokens contained within the same sentence as the entity occurrence and places them to a single document as a continuous sequence. A variation of this strategy introduces a sentence delimiter token, *SENTENCE_SEPARATOR*, to indicate sentence boundaries. One further variation constructs a document containing a continuous sequence of pipe separated lemmas instead of regular tokens. In all variations of our implementation of the *EntityWords* approach the punctuation tokens (full stops, commas and other such tokens) are discarded.

3.4.2 Featurization

Use of supervised machine learning models to classify the generated context documents requires transformation into feature vectors. Multiple approaches to convert text documents into vector space representations were described in section 2.2.2. The previously described method to generate context documents produces documents of widely variable length dependent on the number of times the entity was mentioned within the article. This means that the standard TF-IDF vectorization method [55] may not be expressive enough in case of short documents due to its relatively high dimensionality. As previously hypothesised, usage of pretrained word vectors may enable short documents to better convey sentiment.

In the described implementation, 200 and 100 dimensional *word2vec* word vectors were trialled as part of experiments. Each word corresponds to a single *word2vec* vector. Multiple strategies are available to combine the multiple vectors to a single feature vector that can then be used in a machine learning model. Concatenation would result in rather high dimensional and heterogeneous length vectors and as such is not possible. Averaged and summed feature vector strategies proved more promising and as such were explored during experiments.

3.4.3 Classification approach

Once the preprocessing and featurization steps have been applied to the input text documents, they are used in supervised machine learning models. This means that of the available data set, a portion of the data is regarded as training data and used to train the classifiers. Data not used for training is then used to evaluate the trained model's performance. The current implementation has clear distinctions between the experimentation phase, training a production model phase and the final inference phase.

The experimentation phase is based on the concept of a cross validated experiment. A Python software module providing the necessary scaffolding to run experiments and record results in uniform fashion was developed. An illustration of the uniform experiment process supported by the software module is provided in Figure 6. The module can be modified to enable different combinations of featurization and preprocessing strategies along with different model configurations. The only exception that does not use the developed experiment framework is the experiment to evaluate performance of the ANN based classifier. A minimal variant of the experiment module was ported and used to wrap the Facebook Research's *pytext* NLP framework [56] to support k-fold cross validation.

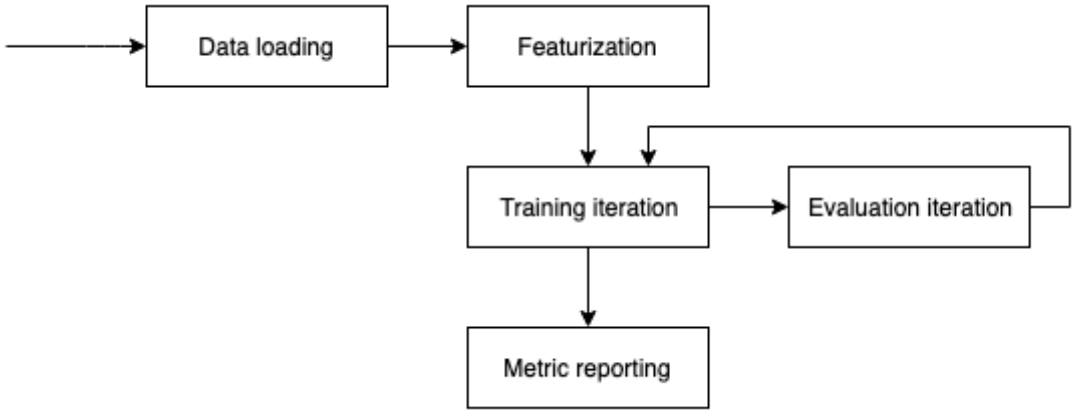

Figure 6: The unified experiment process

Another piece of scaffolding was developed to support the production model training process. The support module was developed to provide an interface encapsulating an unified training process. Only a single training iteration using the whole data set is performed. The resulting model is then serialised into a binary format that can be used to perform inference. A HTTP API wrapping the trained classification model to enable usage by downstream consumers was developed. The API accepts a text document along with output of the entity detection service combined in a JSON encoded object. Preprocessing and featurization is then performed and the API produces polarity ratings for each of the

mentioned entities delivered as a JSON encoded response. Examples of both the request and response formats used by the sentiment classification API are listed in appendix B.

3.4.4 Choice of classification models

Only supervised machine learning models were trialled in experiments. The choice of classification models was based on earlier results achieved by related work into similar problems. In most cases the model implementations available from the *scikit-learn* Python machine learning library [57] were used. Of the more traditional classifiers decision trees [58], SVM [59] and the naïve bayes classification model [60] were used. All those have had previous use in solutions to similar problems [32], [38]. The final two models have also been applied to Estonian language sentiment analysis tasks, with the result achieved using a SVM model counted as state of the art [38]. The *scikit-learn* dummy classifier [61] was used to provide a baseline in classifier performance with predictions based on class distribution.

Of the less commonly used models, the XGBoost [62] model was selected for experiments due to its proven performance in Kaggle competitions. The prebuilt Python package distribution of *libxgboost* was used [63]. An ANN approach was also tried by utilising the capabilities of a new ANN based toolkit for solving NLP problems, open sourced by Facebook Research, called *pytext* [56]. The framework provides a configuration based approach to solve common problems encountered in NLP [64]. Two ANN model implementations for text classification have been implemented in the framework [56]. Of the two, the convolutional neural network based approach [65] was found to perform better in freeform testing with random samples of data from the visible partition of the corpus and as such was used in experiments.

4. Experimental results and discussion

This chapter focuses on experiments performed to evaluate the methodologies described in chapter 3. The first section focuses on how the implemented artefact evaluation process relates to generic information system artefact evaluation strategies. The evaluation methodology of the entity detection artefact is then explored along with final accuracy results achieved the blind partition of the proprietary data set. Common errors and potential improvements to be performed as future work are discussed. Further, the evaluation methodology for the artefact performing ELSA is introduced with differences on the visible and blind data sets. Evaluation results achieved on both portions of the data are provided along with potential improvement strategies and discussion.

4.1 Artefact evaluation criteria

This thesis is focused on the creation of a new potential production system that performs a larger NLP task. This means that the process as a whole can be generalised as information system design. In evaluating the system as a whole, the focus lies on the general system dimension criteria such as efficacy, generality and validity along with environmental dimension's criteria that determine whether the created artefact is consistent with the organisation and technology [66]. The research goals were achieved and software artefacts were delivered to the organisation in a way that enabled immediate use. The design considerations of the system outlined in section 3.1 were designed as such to ensure compliance with the organisational consistency criteria.

Adoption of the service oriented architecture approach with a pipeline based composition improved the delivered artefact's generality. The implemented subsystems may be repurposed as parts of a different NLP problem. Evaluation of the validity and efficacy takes a more experimental focus in our case as the author draws a direct line between these criteria and the activity dimension's accuracy and efficiency criteria. The deployed

artefacts perform tasks for which we can provide experimental performance and accuracy evaluation. The reported experimental accuracy then reflects on the degree of efficacy of the system as a whole.

4.2 Evaluating the entity detection system artefact

The entity detection artefact is a component that is responsible for augmenting an existing NLP system and its NER module with a NEL module backed by an authoritative entity knowledge base. Two main metrics were used to quantify the performance of the artefact: precision, indicating the relevance percentage in discovered entities, and recall, indicating the percentage of correctly discovered entities. Precision is defined as $\frac{T_P}{T_P + F_P}$ where T_P is the true positive rate and F_P the false positive rate [67]. The value of the precision metrics ranges between 0 and 1 (all marked entities were relevant). True positives are defined as detected entities that are included in the annotations and false positives as those that were not noted by human annotators. Using similar notation, recall is then defined as $\frac{T_P}{T_P + F_N}$ where F_N refers to the false negative rate [67]. Similarly, the value ranges between 0 and 1 where the perfect score indicates that all entities mentioned in the article were discovered. In this task false negatives are defined as annotated entities that were not found by the entity detector.

The choice of evaluation metrics was influenced by the business requirements as explained in chapter 3. Although it is important that detected entities have been chosen correctly, false positives are more tolerable than false negatives. It is preferable to ensure that all possible mentions were detected and accept some false positives stemming from uncertainty during the entity linking step over missing mentions present in the article. This leads to recall to be the main evaluation metric and enabled more leniency in the linking steps.

4.2.1 Evaluation methodology

Evaluation of the entity detection artefact on the visible data set was automated. This automation was needed in order to enable continuous performance evaluation during development to gauge the impact of changes to the algorithm. Note that as a rule based approach was used, no further divisions were applied to neither portion of the data set in this case.

The entity detection module produced a set of detected entities that included the fully qualified entity name retrieved from the knowledge base for each. The detected entity set is noted as E_D. Entity annotations were combined to form a single set noted as E_A. Before continuing two terms need to be defined: fuzzy equality comparison and fuzzy set intersection. Fuzzy equality comparison is defined in this case as a form of equality comparison where the *Fuzzywuzzy* library is used to produce a similarity score (in the range of 0 - 100 [68]). If the similarity score is larger than the defined threshold, the two strings are considered equal. Continuing from the previous definition, fuzzy set intersection is then defined as the set intersection operation, with an addition that fuzzy equality comparison is used to find matches between members. These are convenience definitions and deviate from the well known mathematical terms in order to enable conciseness in further definitions.

The true positive set of entities can now be defined for an article as the fuzzy intersection of the detected and annotated entity sets. In the experiments performed on the visible data set, the similarity score threshold was set at 90. The false positive and false negative sets are then defined as $E_D - E_A$ and $E_A - E_D$ based on earlier definitions presented in section 4.2. The source code of the Python script used to perform automated evaluation on the visible portion of the data set is presented in appendix C. The methodology still required some manual preprocessing performed on the annotation data to support the automation. This preprocessing included some normalisation of detected names and manual disambiguation of certain acronyms and colloquial names to their fully qualified legal form.

The earlier definitions also hold in the evaluation process performed on the blind portion of the data set. Given the issues with the corpus annotations that were reported earlier, the blind set evaluation effort was performed manually. The entity detection service was packaged as a ready to use artefact and provided to the evaluator. The evaluator then performed tool assisted processing on the blind set of articles using the artefact. Manual fuzzy comparison was performed between the annotations and discovered entities. Precision and recall metrics were calculated by the evaluator and provided to the author of the thesis.

4.2.2 Achieved results

Recall and precision metrics achieved on the blind partition of the data are reported. The choice of evaluation metrics was explained in section 4.2. Achieved results are listed in Table 3. Section 4.2.3 provides insight into issues affecting the metrics and lists potential future improvements to the implemented methodology.

	Precision	**Recall**
Blind	0.66	0.65

Table 3: Entity detection results

4.2.3 Common errors and limitations

A common form of errors can be classed as mismatches between the human annotator's understanding and the entity knowledge base. Certain annotations referred to corporate entities that are no longer listed. This is due to name changes, mergers and regular business events such as bankruptcies. In the current form, the backing entity knowledge base does not account for these possibilities and is meant to be a snapshot of a specific state of the Estonian Business Register. Similarly, in some cases the annotations contained mentions of colloquial consortia. These are consortia of corporate entities that do not have a namesake entity. Examples of such consortia would be *Alexela Grupp* and *Skinest Grupp*, neither of which has a singular corporate entity heuristically detectable as a representative of the group.

One persistent class of false positives is to do with locations. Street, city and area names often appear as part of entity names registered within the Estonian Business Register. The linking phase does discard too common phrases (more than five matching entities). Wholly discarding all entities classified as a location is not possible as company references may refer to places of business. Some falsely labelled corporate entity mentions would be discarded as well. Possible improvements to eliminate this error are listed below.

4.2.4 Potential future improvements

The entity knowledge base needs to be improved to support historical corporate names and include entities other than currently active ones. Introduction of historical names and entities may also complicate the linking process as after a certain period of time

new corporate entities may register new names similar to those used by defunct corporate entities. This means that more complicated disambiguation logic based on article contents and external facts would be needed. Whether to support colloquial consortia is more of a business decision, but two naïve approaches would be to either construct corresponding entity collections or define a single representative entity and handle it similarly to the colloquial names special case. In case of the first approach, mentions of a consortium would count towards all of its members.

Removal of false positives related to locations could be achieved by including parts of addresses to the blacklist of common phrases described in section 3.3.3. One such extensive list is maintained and published by the Republic of Estonia Land Board [69]. Although due to the size of such a list (the previously mentioned one consists of 1288514 entries at the time of writing) further deliberation is required on how to effectively make use of the export as a linguistic resource.

Including further information in the backing entity knowledge base may also assist in elimination of false positives. One such signal would be the inclusion of economic activity classifiers. The author hypothesises that in general, certain classifiers have higher prior probabilities of appearing together in similar contexts. Other signals could be the reported revenues and numbers of employees — economically active entities should also hypothetically have a higher prior probability of being mentioned. Incorporation of information about physical entities such as board members and shareholders related to corporate entities could further contribute in the false positive elimination process. The probability of the linked entity being a true positive should increase when mentions of related physical entities are detected by the NER module even given the potential prevalence of namesakes. All in all the linking process requires the introduction of a method to encode appearance probability.

Another potential gazzetteer signal that could be incorporated are the article author assigned tags accompanying articles prevalent in online publications. The author proposes another hypothesis that online journalists annotate their articles by tagging relevant entities appearing in articles. The detected and linked entities could then be correlated to lower the true positive probability of detected entities missing from tags. Going even further, a tag ontology could be built by relating the backing entity knowledge base to publication specific tags to avoid trying to match tags and entities by fuzzy string comparison.

4.3 Evaluating the ELSA system artefact

Evaluation of the ELSA performing artefact takes a more experimental approach than the entity detection one. An unified experimentation flow was proposed in section 3.4.3. The unified experiment flow was necessitated by the need to evaluate multiple model and featurisation approach combinations on their performance on the visible set in an uniform fashion. The best performing model was then chosen to be packaged as a production artefact and its accuracy was evaluated on the blind partition of the data set by the Creditinfo evaluator.

4.3.1 Sentiment analysis oriented variant of the proprietary corpus

The visible partition of the proprietary data set introduced in section 3.2 required further preprocessing before it could be used to train and evaluate ELSA capable machine learning models. Separation of annotation information and textual features was proposed. The annotation information was placed in a separate TSV formatted file containing detected entities along with references to the originating article file of the corpus and annotated sentiment class. An example of entries from the annotation information file is provided in Figure 7. The textual feature files contain the generated context documents for the respective entities where each line corresponds to the respective line in the annotation info file. This separation enabled creation of multiple textual portions of the corpus using different context document generation strategies as outlined in section 3.4.1. An example snippet of the textual content, with some content omitted for brevity, is provided in Figure 8.

```
1 file    entity  sentiment_class
2 1.txt {"name": "MONESE LTD EESTI FILIAAL", "occurrence": [["Monese",
    [193, 199]], ["Monese", [890, 896]], ["Monese", [1647, 1653]]], "
    code": "12835419"}  POS
3 0.txt {"name": "EESTI ENERGIA AS", "occurrence": [["Eesti Energia",
    [187, 200]], ["Eesti Energia", [322, 335]]], "code": "10421629"}
    NEG
4 22.txt  {"name": "RIIGIMETSA MAJANDAMISE KESKUS", "occurrence": [["
    Riigimetsa Majandamise Keskuse", [1300, 1330]]], "code": "70004459"
    } NTR
5
```

Figure 7: Example of entries included in the annotation info file of the ELSA oriented corpus

```
1 article_file    context_document    document_word_count
2 1.txt   ... hoolimata Suurbritannia ... tekitatud ebakindlusest
    Euroopas ...  69
3 0.txt Eleon Grupp esitas ... Nelja Energia koondumise menetluses ...
    56
4 22.txt  Suunates see raha hoopis Ida-Virumaa ... looks see selge ...
    32
5
```

Figure 8: Example of entries included in one of the context documents file of the ELSA oriented corpus

Preparation of the sentiment analysis corpus files was automated using Python scripts. The corpus is based on the original annotations provided to the author, but in order to enable automated construction only true positive entities detected by the entity detector were used. Due to the relative inaccuracy of the automated evaluation process this somewhat reduced the availability of training data, but in the current iteration 758 samples were extracted using the automated process. Of the training entries 54 were labelled as $negative$, 243 as $positive$ and 460 as $neutral$. A visual overview of the generated corpus's composition by sentiment class is presented in Figure 9. The majority sentiment class, $neutral$ makes up over half of the samples and as such introduced a noticeable imbalance to the corpus.

4.3.2 Evaluation methodology

Evaluation is again performed on the visible and blind partitions of the *ad hoc* corpus. Different model and featurization approach combination evaluation was automated using the unified experiment framework described in section 3.4.3. Experiments were implemented as child classes of the unified experiment flow base class. Only exception to this was the convolutional neural network based approach which used a lightweight wrapper around the *pytext* toolkit [56] to support the k-fold cross validation process similar to other experiments. All experimental results on the visible partition were gathered over 10 iterations of 10-fold cross validation. The choice of 10 as the k value was motivated by the want to reduce bias and variance in accuracy estimates [70]. Experiments on the blind partition of the data set are again performed manually without cross validation using the packaged production artefact. The achieved results are presented in section 4.3.3.

Metrics recorded as part of the experiments on the visible sentiment analysis corpus in-

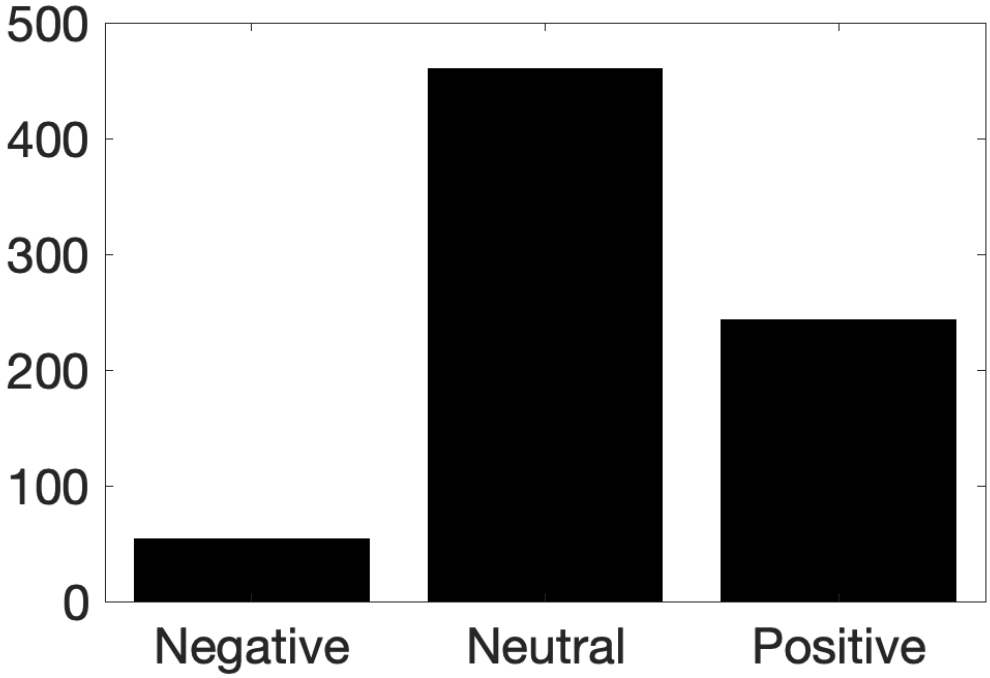

Figure 9: Class composition of the sentiment analysis corpus as a bar chart

cluded the following: accuracy, precision, recall and F_1-score. Calculations were performed using the *scikit-learn* Python library implementations using macro averaging in case of non ANN based approaches. Accuracy was defined as the fraction of correctly predicted test samples or more concisely $\frac{N_{correct}}{N_{samples}}$ [71] where N represents the count. Precision and recall were calculated using macro averaging where the overall metrics correspond to the mean of the per class metrics [71], [72]. In our case, the F_1-score can then be interpreted as the weighted harmonic mean of the macro averaged precision and recall — mathematically expressed as $2 * \frac{P * R}{P + R}$ where P and R stand for macro averaged precision and recall [71], [72]. Note that a class imbalance exists in the data set towards neutral mentions. This bias is acceptable when keeping in mind the potential public facing presence of the data, where incorrect negative classifications may be worse.

4.3.3 Achieved results

Evaluation was performed following the previously described methodology. Experimentation results are reported below in Table 4. The achieved results are segmented by experiment identifier (indicating the model and featurization approach combination) and data partition. Experiment descriptions are listed in Table 5. Metrics that were not recorded as part of any specific experiment are denoted with **-**. Note that all metrics reported on the

visible partition are means of scores reported over a 10 iteration 10-fold cross validated experiment.

Visible partition				
	Accuracy	**Precision**	**Recall**	**F_1-score**
BASELINE	0.48 ± 0.11	0.33 ± 0.12	0.33 ± 0.13	0.33 ± 0.11
DTE200	0.62 ± 0.09	0.51 ± 0.16	0.53 ± 0.18	0.51 ± 0.15
GNBTF-IDF	0.69 ± 0.10	0.61 ± 0.22	0.57 ± 0.16	0.57 ± 0.17
GNBE100	0.61 ± 0.11	0.52 ± 0.14	0.57 ± 0.19	0.52 ± 0.15
GNBE200	0.59 ± 0.10	0.52 ± 0.13	0.57 ± 0.15	0.52 ± 0.12
SVME100	0.74 ± 0.10	0.67 ± 0.34	0.53 ± 0.13	0.54 ± 0.17
SVME200	$\mathbf{0.75 \pm 0.10}$	0.72 ± 0.33	0.54 ± 0.13	0.57 ± 0.18
SVME200 + VC	0.73 ± 0.10	0.65 ± 0.21	$\mathbf{0.58 \pm 0.14}$	$\mathbf{0.59 \pm 0.15}$
SVMB	0.71 ± 0.09	0.79 ± 0.32	0.49 ± 0.13	0.51 ± 0.17
SVME200L	0.73 ± 0.09	$\mathbf{0.74 \pm 0.31}$	0.54 ± 0.14	0.57 ± 0.17
XGBE200	0.73 ± 0.08	0.65 ± 0.26	0.56 ± 0.14	0.58 ± 0.17
CNN	0.71 ± 0.09	-	-	-
Blind partition				
	Accuracy	**Precision**	**Recall**	**F_1-score**
SVME200	0.72	0.46	0.41	0.41

Table 4: Entity-Level Sentiment Analysis results segmented by experiment identifier and data partition

4.3.4 Discussion and potential future improvements

The current work can be considered as more of an initial look into the problem of Estonian language ELSA as multiple potential approaches were trialled without too much focus on model tuning and only a minimal amount of time spent on hyper parameter optimization. Similarly to the results reported in [38] SVM and Naive Bayes classifiers showed promise in Estonian language sentiment analysis tasks. Although both our work and [38] focused on the news domain, direct comparison of the achieved results is not possible due to differences in methodology and task definitions. Based on the results, the original hypothesis that usage of pretrained *word2vec* embeddings may help better carry over sentiment from documents of varying results proved true.

The precision, recall and F_1-score reported on the blind partition in Table 4 are relatively low. The achieved results are better than the random baseline, but not by much. Note that all reported metrics are macro averages. The bias towards the $neutral$ sentiment class became especially apparent during blind evaluation. Class level metrics were all reported as 0 for the $negative$ class in the blind partition. Distinction between the two

Identifier	Description
BASELINE	The *scikit-learn* framework's dummy classifier implementation assigning labels based on class distribution. Provides an evaluation baseline.
DTE200	Decision tree classifier with 200 dimension *word2vec* embeddings
GNBTF-IDF	Gaussian Naive Bayes classifier with TF-IDF vectorization
GNBE100	Gaussian Naive Bayes classifier with 100 dimension *word2vec* embeddings
GNBE200	Gaussian Naive Bayes classifier with 200 dimension *word2vec* embeddings
SVME100	Support vector machine with RBF kernel and 100 dimension *word2vec* embeddings
SVME200	Support vector machine with RBF kernel and 200 dimension *word2vec* embeddings
SVME200 + VC	Support vector machine with RBF kernel, 200 dimension *word2vec* embeddings and added training data from the Estonian valence corpus [40]
SVMB	Support vector machine with RBF kernel and *BERT* generated sentence level embeddings
SVME200L	Support vector machine with RBF kernel and 200 dimension *word2vec* embeddings (lemmatized)
XGBE200	XGBoost classifier with 200 dimension *word2vec* embeddings
CNNE200	Predefined CNN topology [65]. The exact network configuration is listed in appendix D

Table 5: Experiment identifier descriptions

larger classes performed better. For the $positive$ class the precision was 0.65, recall 0.30 and the F_1-score 0.41. As expected, classification performed best on the $neutral$ class with a precision of 0.73, recall of 0.95 and F_1-score of 0.82. The lower precision is due to false positives from other two classes.

Drawing a separation boundary between $positive$ and $negative$ spaces is easier than distinguishing either from the $neutral$ category of texts. Combined with an imbalance towards the $neutral$ class in the training corpus the classifier receives a bias towards the $neutral$ class. Based on some experimentation on the visible partition the author would recommend a different approach over the three class one, possibly by redefining the task at hand in a way that removes the distinction of the $neutral$ class. Combining the $neutral$ and $positive$ would most likely imbalance the corpus further due to the rarity of $negative$ samples, but may result in better detection of $negative$ coverage.

ANN based approaches require more effort in hyper parameter tuning than more traditional supervised machine learning approaches. This is further complicated by their comparatively higher resource requirements during the training phase. Based on the current results, the author hypothesises that given a larger training corpus than the current one ANN based approaches may provide better results in similar tasks. Fine-tuning the *BERT* multi-lingual language model could also be promising given the performance of the classifier making use of *BERT* generated word embeddings.

Another area of improvement in future work would be the adoption of more linguistic information in the context document generation process. The achieved results are rather satisfactory given the simplicity of the currently implemented *EntityWords* approach. In future work, authors should make further use of the estnltk toolkit's dependency parsing functionality. This would enable restriction of context documents to only words that were directly related to mentioned entities. Further rule based features could be defined to, for example, determine whether the more negative implications are towards the entity at hand or another entity. Take for example a sentence announcing a victory in a court case — the sentiment polarity is flipped between the entities.

5. Conclusion

The purpose of this thesis was to create a NLP system solving the problem of corporate entity mention detection along with sentiment analysis in the Estonian language economic news domain. In order to better tackle the larger problem a division based on well known NLP problems was proposed. The entity detection problem was solved as mostly a NEL task using the information from the *estnltk* NER module as one of the signals during entity candidate generation. Reduction to a form of the NEL problem was enabled by the existence of a closed, moderately sized, authoritative entity knowledge base in the form of the Estonian Business Registry.

The second task, sentiment analysis, was reduced to the well known subproblem in sentiment analysis called ELSA. ELSA, Entity-Level Sentiment Analysis, enables the necessary level of sentiment granularity as any article may contain references to multiple entities with varying sentiment towards each. In comparison to regular sentiment analysis further preprocessing was needed to construct *context documents* — sub documents generated from the article which only contain words reflecting sentiment towards a single entity. These documents can be of widely varying length. Previous work in Estonian language sentiment analysis had focused on the paragraph level due to research supporting the paragraph's optimality as a sentiment carrier [38]. Introduction of the *context document* concept made the problem approachable as a generic multiclass document classification problem.

Due to the wide variety in length of context documents, the author hypothesised that making use of word vectors may enable relatively low dimensional vectors to better convey sentiment. Vice versa, relatively short documents could then be turned into vectors better representative of the sentiment they carry instead of high dimensional sparse vectors. Training a custom embedding model was not attempted due to the small size of the available *ad hoc* corpus. *Word2vec* vectors pretrained on the Estonian Koondkorpus [45] and the multilingual *BERT* model [46] were used to generate vector space representations of words in experiments.

Two separate software artefacts were created as part of the work done. Both artefacts offer a JSON over HTTP API for consumption and were written in the Python programming language. Design considerations affecting both software artefacts were discussed in detail in section 3.1. Due to their generality, both artefacts can be repurposed to solve other larger NLP tasks beyond the subject of the thesis at hand. To support a wide variety of deployment environments the final artefacts were packaged as Docker images.

The artefacts were evaluated on a 800 article proprietary *ad hoc* corpus. Further, the process was structured in the form a competition where the corpus was divided into visible and blind partitions. This was done with the aim of preventing potential metric gaming during the evaluation process. Only the larger visible partition of 560 articles was made available to the thesis author. The blind portion was used during final evaluation to perform manual evaluation. Manual evaluation was required due to the annotation process producing human rather than machine readable results.

Evaluation of the entity detection process focused on the precision and recall metrics due to business requirements. The requirements were more permissive towards false positives than false negatives which means that of the two, recall is the more important metric. Due to the adoption of a rule based approach no further division into training and evaluation partitions was applied. Evaluation was manually performed on the blind partition of the corpus and yielded 0.66 precision and 0.65 recall.

Experimental evaluation was used on the sentiment analysis artefact. Multiple classifier and featurization approach combinations were organised into experiments based on an unified experiment framework developed in Python. All experimental metrics reported on the visible partition of the corpus were achieved by performing ten iterations of 10-fold cross validation. The accuracy, precision, recall and the F_1-score performance metrics were gathered. Macro averaging was used for precision, recall and the F_1-score. An imbalance towards neutral sentiment was present in the data set as illustrated in Figure 9.

The best performing featurization and model combination, based on accuracy, was the Support Vector Machine classifier with the RBF kernel trained on 200 dimensional word vectors averaged over the context documents. It achieved 0.75 ± 0.10 accuracy, 0.72 ± 0.33 precision, 0.54 ± 0.13 recall and 0.57 ± 0.18 F_1-score in an experiment performed on the visible partition of the data. Other experiments scoring equal or better F_1-scores were mainly variations of the same SVM based approach and as such keeping in mind the achieved accuracy the model and featurization strategy combination was chosen for manual evaluation on the blind partition. An ANN based approach was also tested, but in the current work did not achieve promising enough results for manual evaluation.

The ELSA artefact evaluation on the blind partition followed the same basic approach as with the entity detection artefact. The packaged artefact was provided to the evaluator and evaluation metrics were provided to the author. The following results were achieved n the blind partition: 0.72 accuracy, 0.46 precision, 0.41 recall and an F_1-score of 0.41. The low precision and recall were influenced by the imbalance present in the training data set towards *neutral* mentions. This lead to a number of false classifications of *negative* and *positive* mentions as *neutral* in the blind partition. Further, during evaluation 0 entries were correctly classified as $negative$ resulting in a noticeable drop in the macro averaged metrics.

Future improvements were discussed for both artefacts separately. In case of the entity detection module the incorporation of further gazzeeteer signals such as the economic activity classifiers, employee counts and revenues was proposed. Addition of Estonian address parts [69] to the common phrases blacklist was also proposed due to their presence being a prominent class of false positive. Introduction of a true positive probability rating was also proposed as it would help better quantify all the different signals contributing to false positive elimination. In sentiment analysis, most future work was recommended to direct focus on hyper parameter and ANN configuration optimisation. Additionally, making further use of the linguistic features to aid in construction of more focused context documents should also improve upon the currently achieved results. Reformulating the task itself to remove the $neutral$ class or otherwise move towards a two class classification problem should also be considered.

In a more general sense future work would be to extend the developed artefacts to support more types of entities. Among such proposed entities for example were copyrights registered with the Estonian Patent Office, made available as open data [73], and addresses of corporate websites. Due to the pipeline approach and preference towards plain text data ingestion in the first step, the data acquisition parts of the complete processing pipeline were left up to the system integrators. Making use of the research into Estonian language machine transcription [74] in data acquisition phase would enable the application of the components developed as part of this thesis on an even wider range of possible mediums. To sum up — the work done in this thesis and the resulting software artefacts are two building blocks required for an intelligent automated media monitoring system for Estonian language news media.

References

[1] S. Orasmaa, T. Petmanson, A. Tkachenko, S. Laur, and H.-J. Kaalep, "Estnltk-nlp toolkit for estonian.", in *LREC*, 2016.

[2] A. Hevner and S. Chatterjee, "Design science research in information systems", in *Design research in information systems*, Springer, 2010, pp. 9–22.

[3] Creditinfo Eesti AS. (2019). About us, [Online]. Available: https://www.creditinfo.ee/en/about-us/ (27.04.2019).

[4] Creditinfo Eesti AS. (2019). E-krediidiinfo, [Online]. Available: https://www.creditinfo.ee/en/e-krediidiinfo-database/ (27.04.2019).

[5] Creditinfo Eesti AS. (2019). Data processing principles, [Online]. Available: https://www.creditinfo.ee/en/about-us/data-processing-principles/ (27.04.2019).

[6] B. Pang, L. Lee, *et al.*, "Opinion mining and sentiment analysis", *Foundations and Trends® in Information Retrieval*, vol. 2, no. 1–2, pp. 1–135, 2008.

[7] A. Mengelkamp, S. Hobert, and M. Schumann, "Corporate credit risk analysis utilizing textual user generated content-a twitter based feasibility study.", in *PACIS*, 2015, p. 236.

[8] R. Grishman and B. Sundheim, "Message understanding conference-6: A brief history", in *Proceedings of the 16th Conference on Computational Linguistics - Volume 1*, ser. COLING '96, Copenhagen, Denmark: Association for Computational Linguistics, 1996, pp. 466–471. DOI: 10.3115/992628.992709. [Online]. Available: https://doi.org/10.3115/992628.992709.

[9] S. Sekine, *Named entity: History and future*, 2004.

[10] A. Borthwick and R. Grishman, "A maximum entropy approach to named entity recognition", PhD thesis, Citeseer, 1999.

[11] C. D. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. J. Bethard, and D. McClosky, "The Stanford CoreNLP natural language processing toolkit", in *Association for Computational Linguistics (ACL) System Demonstrations*, 2014, pp. 55–60. [Online]. Available: http://www.aclweb.org/anthology/P/P14/P14-5010.

[12] M. Honnibal and I. Montani, "Spacy 2: Natural language understanding with bloom embeddings, convolutional neural networks and incremental parsing", *To appear*, 2017.

[13] A. Tkachenko, T. Petmanson, and S. Laur, "Named entity recognition in estonian", in *Proceedings of the 4th Biennial International Workshop on Balto-Slavic Natural Language Processing*, 2013, pp. 78–83.

[14] A. Tkachenko, "Named entity recognition for the estonian language", Master's thesis, 2010.

[15] E. F. T. K. Sang and F. D. Meulder, *Introduction to the conll-2003 shared task: Language-independent named entity recognition*, 2003. arXiv: cs/0306050 [cs.CL].

[16] A. Baevski, S. Edunov, Y. Liu, L. Zettlemoyer, and M. Auli, *Cloze-driven pretraining of self-attention networks*, 2019. arXiv: 1903.07785 [cs.CL].

[17] G. Szarvas, R. Farkas, L. Felföldi, A. Kocsor, and J. Csirik, "A highly accurate named entity corpus for hungarian.", in *LREC*, 2006, pp. 1957–1960.

[18] E. Simon, "Approaches to hungarian named entity recognition", PhD thesis, 2013.

[19] The estnltk authors. (Dec. 6, 2016). Training custom models, [Online]. Available: https://estnltk.github.io/estnltk/1.4.1/tutorials/ner.html%5C#training-custom-models (27.04.2019).

[20] W. Shen, J. Wang, and J. Han, "Entity linking with a knowledge base: Issues, techniques, and solutions", *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, no. 2, pp. 443–460, 2015.

[21] Centre of Registers and Information Systems. (Oct. 1, 2018). Äriregister ning mittetulundusühingute ja sihtasutuste register tegevusalati seisuga 01.01.2019, [Online]. Available: https://www2.rik.ee/rikstatfailid/failid/tabel.php?url=19_01tg.htm (27.04.2019).

[22] E. Fredkin, "Trie memory", *Commun. ACM*, vol. 3, no. 9, pp. 490–499, Sep. 1960, ISSN: 0001-0782. DOI: 10.1145/367390.367400. [Online]. Available: http://doi.acm.org/10.1145/367390.367400.

[23] P. Weiner, "Linear pattern matching algorithms", in *14th Annual Symposium on Switching and Automata Theory (swat 1973)*, Oct. 1973, pp. 1–11. DOI: 10.1109/SWAT.1973.13.

[24] R. Plutchik, "A general psychoevolutionary theory of emotion", in *Theories of emotion*, Elsevier, 1980, pp. 3–33.

[25] S. M. Mohammad, F. Bravo-Marquez, M. Salameh, and S. Kiritchenko, "Semeval-2018 Task 1: Affect in tweets", in *Proceedings of International Workshop on Semantic Evaluation (SemEval-2018)*, New Orleans, LA, USA, 2018.

[26] B. Gokulakrishnan, P. Priyanthan, T. Ragavan, N. Prasath, and A. Perera, "Opinion mining and sentiment analysis on a twitter data stream", in *International Conference on Advances in ICT for Emerging Regions (ICTer2012)*, Dec. 2012, pp. 182–188. DOI: 10.1109/ICTer.2012.6423033.

[27] J. M. Wiebe, "Tracking point of view in narrative", *Comput. Linguist.*, vol. 20, no. 2, pp. 233–287, Jun. 1994, ISSN: 0891-2017. [Online]. Available: http://dl.acm.org/citation.cfm?id=972525.972529.

[28] V. Bobicev and M. Sokolova, "Inter-annotator agreement in sentiment analysis: Machine learning perspective", in *RANLP*, 2017.

[29] R. Artstein and M. Poesio, "Inter-coder agreement for computational linguistics", *Computational Linguistics*, vol. 34, no. 4, pp. 555–596, 2008. DOI: 10.1162/coli.07-034-R2. eprint: https://doi.org/10.1162/coli.07-034-R2. [Online]. Available: https://doi.org/10.1162/coli.07-034-R2.

[30] M. Pontiki, D. Galanis, J. Pavlopoulos, H. Papageorgiou, I. Androutsopoulos, and S. Manandhar, "Semeval-2014 task 4: Aspect based sentiment analysis", in *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, Dublin, Ireland: Association for Computational Linguistics and Dublin City University, Aug. 2014, pp. 27–35. [Online]. Available: http://www.aclweb.org/anthology/S14-2004.

[31] M. Pontiki, D. Galanis, H. Papageorgiou, I. Androutsopoulos, S. Manandhar, A.-S. Mohammad, M. Al-Ayyoub, Y. Zhao, B. Qin, O. De Clercq, *et al.*, "Semeval-2016 task 5: Aspect based sentiment analysis", in *Proceedings of the 10th international workshop on semantic evaluation (SemEval-2016)*, 2016, pp. 19–30.

[32] K. Schouten and F. Frasincar, "Survey on aspect-level sentiment analysis", *IEEE Transactions on Knowledge and Data Engineering*, vol. 28, no. 3, pp. 813–830, 2016.

[33] H. Saif, Y. He, M. Fernandez, and H. Alani, "Contextual semantics for sentiment analysis of twitter", *Information Processing & Management*, vol. 52, no. 1, pp. 5–19, 2016.

[34] A. Esuli and F. Sebastiani, "Sentiwordnet: A publicly available lexical resource for opinion mining.", in *LREC*, Citeseer, vol. 6, 2006, pp. 417–422.

[35] A. Devitt and K. Ahmad, "Sentiment polarity identification in financial news: A cohesion-based approach", in *Proceedings of the 45th annual meeting of the association of computational linguistics*, 2007, pp. 984–991.

[36] N. Godbole, M. Srinivasaiah, and S. Skiena, "Large-scale sentiment analysis for news and blogs.", *Icwsm*, vol. 7, no. 21, pp. 219–222, 2007.

[37] D. Gomez-Zara, M. Boon, and L. Birnbaum, "Who is the hero, the villain, and the victim?: Detection of roles in news articles using natural language techniques", in *23rd International Conference on Intelligent User Interfaces*, ACM, 2018, pp. 311–315.

[38] H. Pajupuu, R. Altrov, and J. Pajupuu, "Identifying polarity in different text types", *Folklore. Electronic Journal of Folklore*, no. 64, pp. 126–138, 2016.

[39] B. Ojamaa, P. K. Jokinen, and K. Muischenk, "Sentiment analysis on conversational texts", in *Proceedings of the 20th Nordic Conference of Computational Linguistics, NODALIDA 2015, May 11-13, 2015, Vilnius, Lithuania*, Linköping University Electronic Press, 2015, pp. 233–237.

[40] Institute of the Estonian Language. (2016). Valence corpus:query, [Online]. Available: `http://peeter.eki.ee:5000/valence/paragraphsquery` (27.04.2019).

[41] G. Jaanimäe, "Eesti wordnet ja meelestatuse analüüs", Master's thesis, 2018.

[42] P. C. Lane, D. Clarke, and P. Hender, "On developing robust models for favourability analysis: Model choice, feature sets and imbalanced data", *Decision Support Systems*, vol. 53, no. 4, pp. 712–718, 2012.

[43] J. Ramos *et al.*, "Using tf-idf to determine word relevance in document queries", in *Proceedings of the first instructional conference on machine learning*, vol. 242, 2003, pp. 133–142.

[44] T. Mikolov, K. Chen, G. Corrado, and J. Dean, *Efficient estimation of word representations in vector space*, 2013. arXiv: `1301.3781 [cs.CL]`.

[45] The estnltk authors. (Mar. 8, 2016). Estnltk/word2vec-models: Word2vec models trained on an estonian text corpus., [Online]. Available: `https://github.com/estnltk/word2vec-models` (27.04.2019).

[46] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding", *arXiv preprint arXiv:1810.04805*, 2018.

[47] Amazon Web Services, Inc. (2019). Aws lambda – serverless compute - amazon web services, [Online]. Available: https://aws.amazon.com/lambda/ (06.04.2019).

[48] Docker Inc. (2019). About docker ce | docker documentation, [Online]. Available: https://docs.docker.com/install/#supported-platforms (06.04.2019).

[49] Kaggle Inc. (2019). Competitions documentation | kaggle, [Online]. Available: https://www.kaggle.com/docs/competitions#competition-formats (18.04.2019).

[50] Republic of Estonia Ministry of Justice. (Jul. 21, 2018). Ärinimi, mittetulundusühingu ja sihtasutuse nimi | justiitsministeerium, [Online]. Available: https://www.just.ee/et/eesmargid-tegevused/praktilisi-nouandeid/arinimi-mittetulundusuhingu-ja-sihtasutuse-nimi (06.04.2019).

[51] Centre of Registers and Information systems. (2019). Äriregistri teabesüsteem, [Online]. Available: https://ariregister.rik.ee/index?lang=eng (09.04.2019).

[52] Google Inc. (Aug. 9, 2017). Google/pygtrie: Python library implementing a trie data structure., [Online]. Available: https://github.com/google/pygtrie (06.04.2019).

[53] H.-J. Kaalep and K. Muischnek. (Jan. 4, 2019). Eesti kirjakeele sagedussõnastik, [Online]. Available: https://www.cl.ut.ee/ressursid/sagedused/index.php?lang=et (27.04.2019).

[54] Eesti Keele Instituut. (2013). Eesti keele instituut - tarkvara, [Online]. Available: https://www.eki.ee/tarkvara/wordlist/ (27.04.2019).

[55] Scikit-learn authors. (Mar. 2019). Sklearn.feature_extraction.text.tfidfvectorizer — scikit-learn 0.20.3 documentation, [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html (27.04.2019).

[56] Facebook Research. (2018). Facebookresearch/pytext: A natural language modeling framework based on pytorch, [Online]. Available: https://github.com/facebookresearch/pytext (09.04.2019).

[57] 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, "Scikit-learn: Machine learning in Python", *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[58] Scikit-learn authors. (Mar. 2019). 1.10. decision trees — scikit-learn 0.20.3 documentation, [Online]. Available: https://scikit-learn.org/stable/modules/tree.html (27.04.2019).

[59] Scikit-learn authors. (Mar. 2019). 1.4. support vector machines — scikit-learn 0.20.3 documentation, [Online]. Available: https://scikit-learn.org/stable/modules/svm.html (27.04.2019).

[60] Scikit-learn authors. (Mar. 2019). 1.9. naive bayes — scikit-learn 0.20.3 documentation, [Online]. Available: https://scikit-learn.org/stable/modules/naive_bayes.html (27.04.2019).

[61] Scikit-learn authors. (Mar. 2019). Sklearn.dummy.dummyclassifier — scikit-learn 0.20.3 documentation, [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.dummy.DummyClassifier.html (27.04.2019).

[62] T. Chen and C. Guestrin, "Xgboost", *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '16*, 2016. DOI: 10.1145/2939672.2939785. [Online]. Available: http://dx.doi.org/10.1145/2939672.2939785.

[63] The XGBoost contributors. (Mar. 5, 2016). Training custom models, [Online]. Available: https://xgboost.readthedocs.io/en/latest/index.html (27.04.2019).

[64] Facebook Research. (2019). Pytext documentation — pytext documentation, [Online]. Available: https://pytext-pytext.readthedocs-hosted.com/en/latest/index.html (09.04.2019).

[65] Y. Kim, "Convolutional neural networks for sentence classification", *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014. DOI: 10.3115/v1/d14-1181. [Online]. Available: http://dx.doi.org/10.3115/v1/D14-1181.

[66] N. Prat, I. Comyn-Wattiau, and J. Akoka, "Artifact evaluation in information systems design-science research-a holistic view.", in *PACIS*, 2014, p. 23.

[67] D. Olson and D. Delen, *Advanced Data Mining Techniques*. Springer Berlin Heidelberg, 2008, ISBN: 9783540769170. [Online]. Available: https://books.google.ee/books?id=2vb-LZEn8uUC.

[68] Seatgeek. (Aug. 20, 2018). Seatgeek/fuzzywuzzy: Fuzzy string matching in python, [Online]. Available: https://github.com/seatgeek/fuzzywuzzy (06.04.2019).

[69] Republic of Estonia Land Board. (Apr. 2019). X-gis(3) portal, [Online]. Available: http://xgis.maaamet.ee/adsavalik/ads (20.04.2019).

[70] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An Introduction to Statistical Learning: with Applications in R*, ser. Springer Texts in Statistics. Springer New York, 2013, ISBN: 9781461471387. [Online]. Available: https://books.google.ee/books?id=qcI%5C_AAAAQBAJ.

[71] Scikit-learn authors. (Mar. 2019). 3.3. model evaluation: Quantifying the quality of predictions — scikit-learn 0.20.3 documentation, [Online]. Available: https://scikit-learn.org/stable/modules/model_evaluation.html (27.04.2019).

[72] M. Sokolova and G. Lapalme, "A systematic analysis of performance measures for classification tasks", *Information Processing & Management*, vol. 45, no. 4, pp. 427–437, 2009, ISSN: 0306-4573. DOI: https://doi.org/10.1016/j.ipm.2009.03.002. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0306457309000259.

[73] Estonian Patent Office. (Oct. 1, 2018). Patendiameti avalikud andmed, [Online]. Available: https://opendata.riik.ee/andmehulgad/eesti-patendiameti-kaubam-rkide-avalikud-andmed/ (27.04.2019).

[74] T. Alumäe. (Jan. 8, 2019). Alumae/kaldi-offline-transcriber: Offline transcription system for estonian using kaldi, [Online]. Available: https://github.com/alumae/kaldi-offline-transcriber (06.04.2019).

A. Example of the entity detection API response


```
{
    "11488826": {
        "name": "DANSKE BANK A/S EESTI FILIAAL",
        "occurrence": [
            [
                "Danske Bank",
                [
                    425,
                    436
                ]
            ]
        ]
    },
    "70000272": {
        "name": "RAHANDUSMINISTEERIUM",
        "occurrence": [
            [
                "rahandusministeeriumile",
                [
                    3845,
                    3868
                ]
            ],
            [
                "rahandusministeerium",
                [
                    4199,
                    4219
                ]
            ]
        ]
```

```
    },
    "70000898": {
        "name": "JUSTIITSMINISTEERIUM",
        "occurrence": [
            [
                "justiitsministeeriumiga",
                [
                    4242,
                    4265
                ]
            ]
        ]
    },
    "74000174": {
        "name": "EESTI PANK",
        "occurrence": [
            [
                "Eesti Panga",
                [
                    3638,
                    3649
                ]
            ],
            [
                "Eesti Panga",
                [
                    3664,
                    3675
                ]
            ]
        ]
    },
    "80052459": {
        "name": "SOTSIAALDEMOKRAATLIK ERAKOND",
        "occurrence": [
            [
                "SDE",
                [
                    4169,
                    4172
                ]
            ]
        ]
    }
}
```

ELSA API response

B. Examples of ELSA artefact API request and response


```
{
"text": "Full text omitted for brevity purposes",
"entities":{
    "11488826": {
        "name": "DANSKE BANK A/S EESTI FILIAAL",
        "occurrence": [
            [
                "Danske Bank",
                [
                    425,
                    436
                ]
            ]
        ]
    },
    "70000272": {
        "name": "RAHANDUSMINISTEERIUM",
        "occurrence": [
            [
                "rahandusministeeriumile",
                [
                    3845,
                    3868
                ]
            ],
            [
                "rahandusministeerium",
                [
                    4199,
                    4219
                ]
```

```
            ]
        ]
    },
    "70000898": {
        "name": "JUSTIITSMINISTEERIUM",
        "occurrence": [
            [
                "justiitsministeeriumiga",
                [
                    4242,
                    4265
                ]
            ]
        ]
    },
    "74000174": {
        "name": "EESTI PANK",
        "occurrence": [
            [
                "Eesti Panga",
                [
                    3638,
                    3649
                ]
            ],
            [
                "Eesti Panga",
                [
                    3664,
                    3675
                ]
            ]
        ]
    },
    "80052459": {
        "name": "SOTSIAALDEMOKRAATLIK ERAKOND",
        "occurrence": [
            [
                "SDE",
                [
                    4169,
                    4172
                ]
            ]
        ]
```

```
77       }
78 }
79 }
```

ELSA API request

```
1 {"11488826": "neutral", "70000272": "neutral", "70000898": "neutral", "
    74000174": "neutral", "80052459": "neutral"}
```

ELSA API response

C. Automated evaluation script for the entity detector on the visible data set

The script below performs an automated evaluation of entity detection performance given the existence of both annotation and data files.

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

import csv

from entity_detector.detector import detect_registered_entities
import estnltk
from fuzzywuzzy import fuzz

f = open('articles_train.txt', 'r')

reader = csv.reader(f, delimiter='\t')
next(reader)


def fuzzy_in(item, container):
    return any(filter(lambda i: fuzz.token_set_ratio(i, item) >= 90,
    container))


print('file', 'true_positive_count', 'false_positive_count', '
    false_negative_count', 'total', 'accuracy', 'recall',
       'precision', 'f1', 'true_positives', 'false_positives', '
    false_negatives', sep='\t')
total_tp_count = 0
total_fp_count = 0
total_fn_count = 0
total_entities = 0
```

```
26
27 for i, line in enumerate(reader):
28     pos = [s.strip() for s in line[3].split(',') if len(s.strip()) > 0]
29     neg = [s.strip() for s in line[4].split(',') if len(s.strip()) > 0]
30     neutral = [s.strip() for s in line[5].split(',') if len(s.strip())
       > 0]
31
32     annotated_entities = pos + neg + neutral
33
34     # Read the article & detect the registered entities
35     with open('./data/ap-set/{}'.format(line[2]), 'r') as af:
36         entities = detect_registered_entities(estnltk.Text(af.read()))
37         detected_entities = [e['name'] for e in entities.values()]
38
39     true_positives = set([e for e in detected_entities if fuzzy_in(e,
       annotated_entities)])
40     false_positives = set(detected_entities).difference(true_positives)
41     false_negatives = set([e for e in annotated_entities if not
       fuzzy_in(e, detected_entities)])
42
43     total = len(true_positives) + len(false_positives) + len(
       false_negatives)
44     accuracy = len(true_positives) / total if total > 0 else 0
45     recall = len(true_positives) / (len(true_positives) + len(
       false_negatives)) if len(true_positives) > 0 else 0
46     precision = len(true_positives) / (len(true_positives) + len(
       false_positives)) if len(true_positives) > 0 else 0
47     f1 = 2 * ((precision * recall) / (precision + recall)) if precision
        + recall > 0 else 0
48
49     total_tp_count += len(true_positives)
50     total_fp_count += len(false_positives)
51     total_fn_count += len(false_negatives)
52     total_entities += total
53
54     print(line[2], len(true_positives), len(false_positives), len(
       false_negatives), total, accuracy, recall, precision,
55           f1, true_positives, false_positives, false_negatives, sep='\t
       ')
56
57
58 accuracy = total_tp_count / total_entities
59 recall = total_tp_count / (total_tp_count + total_fn_count)
60 precision = total_tp_count / (total_tp_count + total_fp_count)
61 f1 = 2 * ((precision * recall) / (precision + recall))
```

```
62 print('total', total_tp_count, total_fp_count, total_fn_count,
       total_entities, accuracy, recall, precision,
63         f1, {}, {}, {}, sep='\t')
64 f.close()
```

Python script for automated evaluation on the visible dataset

D. Pytext configuration for the CNN based classifier

```
{
  "debug_path": "/tmp/model.debug",
  "distributed_world_size": 1,
  "export_caffe2_path": "/tmp/model.caffe2.predictor",
  "export_onnx_path": "/tmp/model.onnx",
  "load_snapshot_path": "",
  "modules_save_dir": "",
  "save_module_checkpoints": false,
  "save_snapshot_path": "/tmp/model.pt",
  "task": {
    "DocClassificationTask": {
      "data_handler": {
        "columns_to_read": [
          "doc_label",
          "text"
        ],
        "eval_batch_size": 10,
        "eval_path": "pytext_sentiment_training_complete.tsv",
        "max_seq_len": -1,
        "shuffle": true,
        "sort_within_batch": true,
        "test_batch_size": 10,
        "test_path": "pytext_sentiment_training_complete.tsv",
        "train_batch_size": 10,
        "train_path": "pytext_sentiment_training_complete.tsv"
      },
      "exporter": null,
      "features": {
        "char_feat": null,
        "dense_feat": null,
        "dict_feat": null,
```

```
        "freeze": false,
        "load_path": null,
        "pretrained_model_embedding": null,
        "save_path": null,
        "shared_module_key": null,
        "word_feat": {
          "embed_dim": 200,
          "embedding_init_range": null,
          "embedding_init_strategy": "zero",
          "export_input_names": [
            "tokens_vals"
          ],
          "freeze": false,
          "lowercase_tokens": false,
          "min_freq": 1,
          "mlp_layer_dims": [],
          "pretrained_embeddings_path": "Kr_lect_1a",
          "vocab_file": "",
          "vocab_from_all_data": false,
          "vocab_from_pretrained_embeddings": false,
          "vocab_from_train_data": true,
          "vocab_size": 0
        }
      },
      "featurizer": {
        "SimpleFeaturizer": {
          "convert_to_bytes": false,
          "lowercase_tokens": false,
          "sentence_markers": null,
          "split_regex": "\\s+"
        }
      },
      "labels": {
        "export_output_names": [
          "doc_scores"
        ],
        "label_weights": {},
        "target_prob": false
      },
      "metric_reporter": {
        "output_path": "/tmp/test_out.txt"
      },
      "model": {
        "decoder": {
          "freeze": false,
```

```
        "hidden_dims": [],
        "load_path": null,
        "save_path": null,
        "shared_module_key": null
      },
      "output_layer": {
        "freeze": false,
        "load_path": null,
        "loss": {
          "CrossEntropyLoss": {}
        },
        "save_path": null,
        "shared_module_key": null
      },
      "representation": {
        "DocNNRepresentation": {
          "cnn": {
            "kernel_num": 100,
            "kernel_sizes": [2, 4, 5]
          },
          "dropout": 0.5
        }
      }
    },
    "optimizer": {
      "Adam": {
        "lr": 0.01,
        "weight_decay": 0.001
      }
    },
    "scheduler": {
      "T_max": 1000,
      "cooldown": 0,
      "cut_frac": 0.1,
      "eta_min": 0,
      "gamma": 0.1,
      "lm_gradual_unfreezing": true,
      "lm_lr_multiplier": 1.0,
      "lm_use_per_layer_lr": false,
      "non_pretrained_param_groups": 2,
      "patience": 5,
      "ratio": 32,
      "step_size": 30,
      "threshold": 0.0001,
      "threshold_is_absolute": false,
```

```
        "type": "none"
      },
      "trainer": {
        "early_stop_after": 0,
        "epochs": 20,
        "max_clip_norm": null,
        "random_seed": 0,
        "report_train_metrics": false
      }
    }
  },
  "test_out_path": "/tmp/test_out.txt",
  "use_cuda_if_available": true,
  "use_tensorboard": false
}
```

Neural network configuration used to perform sentiment analysis on the visible partition of the data