School of Computer Science

Named Entity Recogniser – An Icelandic Prototype

February 2019

BSc Computer Science

Named Entity Recogniser – An Icelandic Prototype

Sigurjón Þorsteinsson & Svanhvít Lilja Ingólfsdóttir

February 2019

TÖLVUNARFRÆÐIDEILD
SCHOOL OF COMPUTER SCIENCE

Named Entity Recogniser – An Icelandic Prototype

Sigurjón Þorsteinsson
Reykjavik University
sigurjont@ru.is

Svanhvít Lilja Ingólfsdóttir
Reykjavik University
svanhviti16@ru.is

Útdráttur

Nafnaþekkjari finnur og flokkar sérnöfn í texta. Hann er eitt af grunnverkfærum máltækni, einkum við þróun hugbúnaðar til upplýsingaútdráttar. Hér er kynnt til sögunnar frumgerð að íslenskum nafnaþekkjara sem er útfærður með gervitauganeti. Forsenda þjálfunar á slíkum netum er að til sé málheild þar sem sérnöfn eru auðkennd og rétt flokkuð. Slík þjálfunarmálheild hefur ekki verið til fyrir íslensku og var gerð hennar hluti af þessu verkefni. Við útfærslu nafnaþekkjarans var notuð tilbúin lausn sem kallast NeuroNER og er sérstaklega hönnuð með sérnafnaflokkun í huga. Niðurstöðurnar benda til þess að þetta sé raunhæf aðferð til að greina sérnöfn í íslensku (F_1=81,3%), sérstaklega með tilliti til þess að þjálfunarmálheildin er ekki stór. Orðavigrar búnir til úr mun stærri málheild reyndust bæta niðurstöðurnar mjög, og eru verðugt rannsóknarefni.

Abstract

A named entity recogniser finds named entities (proper nouns) in a text, and labels them by category. It is a fundamental tool in natural language processing, in particular in the development of information extraction systems. In this paper, we present a prototype of a named entity recogniser for Icelandic, based on artificial neural networks. The training of such networks requires a textual corpus where named entities have been labelled. As no such corpus exists for Icelandic, its creation is a subject of this project. The recogniser was built using NeuroNER, a software package designed for named entity recognition. The results indicate that this is a viable approach towards recognition of named entities in Icelandic (F_1=81.3%), especially considering the moderate size of the training corpus. Word embeddings, created from a much larger unlabelled corpus, turned out to improve the results greatly, warranting further study.

1 Introduction

Natural language processing (NLP) is a research area that addresses how humans and computers can interact using human language. This involves research on how computers can be made to understand and use text or speech, which implies gathering thorough understanding of natural languages and how humans use them (Chowdhury 2003). NLP is an interdisciplinary field with roots in different research areas, such as Computer Science, Linguistics, Mathematics, Engineering, Data Science, Anthropology and Psychology.

A particular subfield of NLP is information extraction, the task of extracting structured data from unstructured texts. In recent years, the amount of data available online has exploded, opening up many possibilities of gathering insight into various aspects. This data, however, is of limited use in its unstructured form. Information extraction deals with this problem by creating databases of useful information extracted from raw texts.

Named entity recognition (NER), a particular subtask of information extraction, is the focus of this paper. NER aims at identifying named entities in texts written in a natural language. These are the entities of text that refer to particular beings, places, organisations and other entities that have been given a name.

NER is a fundamental component of various NLP systems, such as in question answering, relation extraction, machine translation, and sentiment analysis. NER can be useful to different

organisations, for classification of content, faster text search, and deletion of sensitive personal data from documents, as well as for extracting information in biomedicine and other specialised fields.

In this paper, we present a prototype of an Icelandic general purpose named entity recogniser and a labelled training corpus. The NER model was created using NeuroNER (Dernoncourt, Lee, and Szolovits 2017), a NER program based on Artificial Neural Networks (ANN), and defines four entity types: persons (PER), locations (LOC), organisations (ORG) and miscellaneous (MISC).

To our knowledge, this is the first Icelandic named entity recogniser that uses machine learning (ML), and the first annotated training corpus especially developed for the purposes of NER in Icelandic.

Firstly, this paper briefly describes previous work in the field of NER and the state-of-the-art technology. Secondly, we discuss the definition and classification of named entities in Icelandic and give an overview of the methods and data sets used in this implementation. A subsection is dedicated to ANNs and the specific variants that concern this paper. This subsection also introduces word embeddings, and how they can be used to improve the performance of named entity recognisers. Thirdly, the experiment itself is described and finally the results are evaluated and discussed.

2 Related work

The term *named entity* emerged in the 1990s when it was coined for the sixth Message Understanding Conference held in 1995 (Grishman and Sundheim 1996). The term's purpose was to identify the names of people, organisations and geographic locations mentioned in a text, and has since been expanded to include various other entity types, as well as many temporal and numeral expressions (Nadeau and Sekine 2007).

The development of NER has followed a pattern similar to many other fields of NLP, in that the first NER programs were built using handcrafted rules, then various ML methods came along, hybrid systems were developed, and recent research mainly focuses on ANNs as a basis for named entity recognisers. In this chapter, we will give a short overview of the history of NER, and the previous research on the subject for the Icelandic language.

2.1 Rule-based systems

From its inception there have been various approaches to NER, and many rule-based and ML NER programs have been developed. Early on, research was focused on systems based on handcrafted rules and stochastic methods (Rau 1991, Riloff and Phillips 2004). Most of the rule-based systems relied on the following three components: a set of linguistic rules for detecting named entities, gazetteers (lists) containing different types of named entities, and an engine that applies these rules and gazetteers to a text, to recognise and categorise the named entities. Rule-based systems can achieve relatively high precision with carefully engineered features[1] and they don't require a labelled corpus (Mohit 2014). The drawback is that designing the features is time-consuming and requires expertise, and these systems don't handle unknown entities well. This particular issue is a disadvantage in NER, as it is the nature of named entities that new ones appear every day. This method can however yield good results in well-defined domains where there is less variance and an exhaustive lexicon is available, such as in biomedicine (Chiticariu et al. 2010).

2.2 ML methods

ML methods took over as the main approach to NER as the field developed and data became more readily available. These range from unsupervised to fully supervised methods, as well as hybrid systems that rely partly on rules and gazetteers.

Unsupervised learning methods try to model the underlying structure or distribution in the data using the data itself. In NER, unsupervised methods use different algorithms to extract the named entities from an unlabelled corpus, incorporating sets of rules and gazetteers (Collins and Singer 1999, Nadeau, Turney, and Matwin 2006, Zhang and Elhadad 2013). Unsupervised methods eliminate the need for labelling a corpus, a time-consuming task, but instead they must incorporate cleverly engineered features to capture the named entities (Nadeau, Turney, and Matwin 2006).

Supervised learning systems for NER, on the other hand, use a labelled training corpus, where all named entities have been annotated and categorised. Various supervised learning algorithms

[1] A *feature* in the context of NLP is any measurable property of a text that can be input to a ML model. For further reading, see Jurafsky and Martin 2018 and C. D. Manning, Raghavan, and Schütze 2008.

have been studied and different methods have been tested and implemented. Such algorithms include Decision Trees (Paliouras et al. 2000), Support Vector Machines (Y.-C. Wu et al. 2006), Maximum Entropy Models (Ahmed and R 2015) and Conditional Random Fields (CRF) (Finkel, Grenager, and C. Manning 2005), with CRF being the most popular (Y. Wu et al. 2015). Out of ML systems that don't use ANNs, semi-supervised approaches generally yield the best results, see for example Agerri and Rigau (2017) and Passos, Kumar, and McCallum (2014).

Following the rise in popularity of ANNs, recent research has mainly focused on how they can be used to create better NER systems. NER systems based on ANNs have several advantages over rule-based and statistical methods for NER. They deal well with unknown words and don't rely on handcrafted features, unlike the above mentioned methods. They do however need substantially larger annotated training sets than other supervised ML methods, and are more computationally expensive (Seif 2018).

One of the first ANN architectures for NER was proposed by Collobert and Weston (2008). It was constructed from manually engineered features, but this approach was quickly abandoned and replaced by unsupervised processes.

According to a recent survey of advances in NER models (Yadav and Bethard 2018), ANN approaches to NER can be classified according to how the words fed into the networks are represented. The most popular classes of input representations seen today are:

- Word level architectures using word embeddings
- Combined architectures using both word and character embeddings.

Word embeddings will be discussed further in Section 3.6.3.

ANNs for NER are usually implemented using either Convolutional Neural Networks (Chiu and Nichols 2015) or Long Short-Term Memory (LSTM) Recurrent Neural Networks (RNN) (Huang, Xu, and Yu 2015 and Dernoncourt, Lee, and Szolovits 2017). The approaches relevant to this study are described further in Section 3.6.

2.3 Evaluation metrics

Before continuing the discussion, we must explain the metrics used to evaluate performance of NER systems and other NLP models. *Precision* and *recall*, along with their F_1 harmonic mean, are the most commonly used and are defined as follows (Tjong Kim Sang and De Meulder 2003):

$$Precision = \frac{number\ of\ correct\ predictions}{number\ of\ predictions}$$

$$Recall = \frac{number\ of\ correct\ predictions}{true\ number\ of\ named\ entities}$$

The F_1 score (F_1 hereafter) is defined from precision and recall as follows:

$$F_1 = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

Shared tasks have been conducted on NER, most notably the CoNLL shared tasks from 2002 and 2003 (Tjong Kim Sang and De Meulder 2003). An important product of these tasks were the labelled data sets that are still the benchmark for NER tools for English, German, Spanish and Dutch. Researchers can compare the performance of their work to other previously developed tools by training and testing on these data sets. This is a standard procedure in all major NER systems, not only to know the system's performance in relation to others, but also to get listed in survey articles. Further information on CoNLL can be found in Section 5.

A comprehensive overview of the performance of recently published NER systems on CoNLL data sets is given by Yadav and Bethard (2018). Top-scorers for English reach F_1 around 91,5%, and for German, the top score reported is 79%. This difference can be explained by the fact that German is considerably more morphologically complex than English, and all German nouns are capitalised, resulting in more factors for the systems to consider. The above summary reveals one additional interesting fact; the top-scorers from non-ANN and ANN methods of ML NER achieve about the same F_1 for English.

2.4 NER for Icelandic

In the beginning, NER was mainly focused on English, and most of the research was based on

English corpora, but NER systems have since been developed for many languages.

No labelled data sets, such as CoNLL, exist for Icelandic and many other languages with a limited history of NLP, and annotating a training corpus of a viable size is both costly and time-consuming (Lample et al. 2016). This is probably the reason why no ML NER tools have been developed for Icelandic before, according to our best knowledge. This also means that there is no benchmark for evaluation of new NER implementations.

A rule-based NER exists for Icelandic and is part of the IceNLP natural language processing toolkit (Loftsson and Rögnvaldsson 2007), called IceNER. It has been reported to reach F_1 of 71.5% without querying a gazetteer, and around 79% using a gazetteer (Tryggvason 2009). Another rule-based named entity recogniser called Íslenskur textaskimi was in development in 2006, but no published results were found.

Greynir[2] is an Icelandic open-source NLP toolset and website that processes news from Icelandic news sites and offers various insights into the information contained therein (Vilhjálmur Þorsteinsson 2018). One of the features of Greynir is a rule-based named entity recogniser used to find and label person names in the news texts. Greynir also tries to determine relations in the text, for example between persons and their titles. The performance of this named entity recogniser has not been evaluated.

3 Methods and Materials

This section discusses the different methods, ideas, technologies and data used to produce our NER prototype. Following an introduction on the classification of named entities in Icelandic, we describe the data and methods used to create the training corpus. Then we present an overview of the most important ideas and methods used in NeuroNER.

3.1 Named entities and classification

For the task of recognising and categorising named entities, there needs to be a consensus of what constitutes a named entity, and how they should be categorised. We also need to look at the case of Icelandic in particular. Here we will take a closer look at how this task was defined.

[2] https://greynir.is/

3.1.1 Defining named entities for Icelandic

The concept of a *named entity* does not exist as such in Icelandic, and in the limited literature available in Icelandic on the subject, the terms used are "nafn" (name) and "sérnafn" (proper noun) (Nikulásdóttir, Guðnason, and S. Steingrímsson 2017). Therefore, we would like to clarify how we define the term in this paper.

The official definition of "sérnafn" in Icelandic is "the names of individuals or entities of a particular kind, in particular people and other beings, locations, organisations and various works created by man, such as buildings, artworks and intellectual property. Proper nouns are capitalized."[3] (Íslensk málnefnd 2016, p. 3)

This is the most straightforward definition of a proper noun in Icelandic, taken from the official Icelandic orthography rules from 2016. The same document reveals, however, that there are various exceptions to this rule, as well as in the capitalisation of common nouns. In fact, the first 11 pages of the orthography rules are dedicated to a discussion on when proper and common names should be capitalised and when they should not. These exceptions and special cases are too many to review here, but some examples will be given to explain the intricacies of these rules and the issues they may raise in the case of NER.

Common nouns that are to be capitalised are for example "Bandaríkjaforseti" (President of the United States) and "Morgunblaðsritstjóri" (editor in chief at Morgunblaðið), as they are derived from a proper noun. This rule can however not be generalised, as the same proper noun prefix can be capitalised or not depending on the common base noun, as in the case of "Vínarsáttmálinn" (the Vienna Convention) and "vínarbrauð" (Danish pastry), which both are derived from the city of Vienna. Furthermore, capitalisation is optional in some cases, such as in the words "norðurpóll"/"Norðurpóll" (North Pole) and "guð"/"Guð" (god). Additionally, some proper nouns are by convention not to be capitalised, such as "helvíti" (hell) and "djöfullinn" (the Devil) (Íslensk málnefnd 2016).

While the above are the official spelling rules for Icelandic, it is safe to say that the general population, and even experts, regard proper nouns

[3] "Sérnöfn eru heiti sem tilteknir einstaklingar eða eintök af einhverri tegund bera, einkum menn og aðrar verur, staðir, stofnanir og ýmis mannanna verk, eins og hús, listaverk og hugverk. Sérnöfn eru rituð með stórum upphafsstaf."

as those that are capitalised, and common nouns those that are not, as in the official definition. This was the approach taken in this research as well, and by doing so we follow the Icelandic Gold Standard (Loftsson, Yngvason, et al. 2010, referred to here as the Gold Standard), a part of speech (PoS) tagged and hand-reviewed corpus for Icelandic (see Section 3.2). It is quite strict in tagging only capitalised Icelandic nouns as proper nouns. All foreign words have the same tag, so for these we needed to distinguish between common and proper nouns, using the same convention as with the Icelandic nouns.

Another issue, not restricted to Icelandic, is the unorthodox spelling and lack of proper capitalisation in informal texts, such as blogs and social media posts. This poses a special problem for recognising named entities, as the main feature that most NER methods rely on is the capitalisation of proper nouns. Solutions have been proposed to address this problem, especially in NER implementations using Twitter as their source of lexical data, such as in Ritter et al. (2011) and in the WNUT2017 Shared Task on Novel and Emerging Entity Recognition (2017). As our corpus includes in part informal texts, such as blogs and other unedited data, we need to establish a guideline on how to address uncapitalised proper nouns in the data, such as "melrose place" or "kittý". The Gold Standard we rely on tags these as common nouns, so we used the same approach.

Yet another challenge of NER is deciding on word boundaries of named entities spanning more than one word. This is often easy to infer, such as in person names and most locations, but some cases are not as obvious, such as whether to include company endings ("ehf."), street numbers ("Laugavegur 47") or product names ("Intel Core 2 T5450 13.3"). In these and other ambiguous cases we used heuristics and best judgement to decide on entity boundaries.

It should also be noted that title case is not used in Icelandic ("Sjálfstætt fólk", not "Sjálfstætt Fólk"), and common nouns that are part of a proper noun phrase are not to be capitalised ("Geislavarnir ríkisins", not "Geislavarnir Ríkisins"). This can make it harder for an automatic system to infer boundaries for named entities.

The Gold Standard only tags the capitalised word in each named entity as a proper noun, so to distinguish named entities of more than one word, we used the IOB notation[4], commonly used in NLP to indicate text chunk boundaries (Ramshaw and Marcus 1995).

To sum up, in determining what constitutes a named entity, we generally follow the Gold Standard tagging of a proper noun, which for the purpose of this research is used synonymously with the term "named entity".

3.1.2 Named entity taxonomy

A named entity recogniser not only needs to find named entities in a text. To be of any practical use it also has to be able to categorise the named entities.

As previously mentioned, the first shared task on NER focused on recognising people, organisations, and geographic locations in a text (Grishman and Sundheim 1996). Later shared tasks also included the category "miscellaneous" or "other" (Tjong Kim Sang 2002, Tjong Kim Sang and De Meulder 2003), as well as certain temporal and numeral expressions, i.e. dates, times and prices, both in numbers ("2018") as well as written out in words ("next week") (Ferro et al. 2005). In some specialised domains where NER has proven useful, such as bioinformatics, other entity types have been defined, e.g. protein and gene types (Tanabe et al. 2005).

In our NER model, we opted for the four generic classes: PER, LOC, ORG, and MISC, i.e. entities not belonging to the previous three groups. This was thought to be sufficient for the project's scope and its purpose as a named entity recogniser for general use. This categorisation could later be further enhanced with more detailed entity types, such as can be seen in HFST-SweNER (Kokkinakis et al. 2014), a Swedish implementation, which in addition to the three general categories includes Artifacts (products, prizes, etc.), Work&Art (printed material, films, novels, etc.), Events (religious, athletic, scientific, races, battles, etc.), Measure/Numerical (volume, age, index, speed, etc.), and Temporal (times, dates, temporal expressions).

Following is a more detailed definition for each of the four entity types, used as a guide when creating our training corpus:

- **Persons**: Names of humans and other beings, real or fictional, deities, pet names.

[4] Word labels are tagged as inside, outside, and beginning, denoting the location of the corresponding word within an entity/text chunk.

- **Locations**: Names of locations, real or fictional, i.e. buildings, street and place names, both real and fictional. All geographical and geopolitical entities such as cities, countries, counties and regions, as well as planet names and other outer space entities.
- **Organisations**: Icelandic and foreign companies and other organisations, public or private, real or fictional. Schools, churches, swimming pools, community centres, musical groups, other affiliations.
- **Miscellaneous**: All other capitalised nouns and noun phrases, such as works of art, products, events, printed materials, ships and other named means of transportation, etc.

Heuristics were used to resolve doubts, and even though some of these classifications may be debated, we deemed that the most important factor was to be consistent in the categorisation. This was done by keeping track of these details, and by charging one person with the task of hand-reviewing the resulting corpus. No evaluation of the classification process was carried out, both due to time constraints and the fact that no benchmark corpus exists for annotating named entities for Icelandic.

3.2 Corpus

In order to be able to train an ANN for classification tasks, a substantial set of labelled training data is needed, such as the CoNLL data set. As this kind of a corpus does not exist for NER in Icelandic, we had our work cut out for us. A corpus of Icelandic texts needed to be created, with its named entities labelled according to the types defined in Section 3.1.2.

For a general purpose named entity recogniser, it was preferable to use a balanced corpus, i.e. a collection of different types of texts from various sources, representative of written texts in Icelandic. This kind of textual database exists, and is what we used as the source for our training data, namely, the Gold Standard (Loftsson, Yngvason, et al. 2010) mentioned previously. It is an Icelandic PoS tagged corpus, run and maintained by Árni Magnússon Institute for Icelandic Studies (Árnastofnun). The corpus contains approximately 1 million tokens[5] of texts from 13 different sources, written in 2000-2010. These are texts from various online content, along with news from papers, radio and TV; books, laws, elementary school essays, adjudications, and speeches. The texts have been tokenised and PoS tagged automatically using the tagset developed for the Icelandic Frequency Dictionary corpus (described in Helgadóttir, Loftsson, and Rögnvaldsson 2014), with subsequent manual corrections. The F-score of the Gold Standard has been estimated approximately 93%.

Version 1.0 of the Gold Standard was released when the project was in its initial stages. This version included lemmas, i.e. the dictionary forms of the tokens. Those lemmas were created automatically using the Icelandic lemmatiser *Nefnir*[6] (Daðason 2018). The lemmas (see last column in Fig. 1) hadn't been manually reviewed and have now been removed from the online source. This lemmatised version is the one used in this work.

```
Illmennska  nven    illmennska
Holbergs    nkee-s  Holberg
er  sfg3en  vera
meðfædd lvensf  meðfæddur
og  c   og
vandkvæðum  nhfþ    vandkvæði
bundið  sþghen  binda
að  cn  að
hafa    sng hafa
hemil   nkeo    hemill
á   aþ  á
slíkum  fbkfþ   slíkur
brotamönnum nkfþ    brotamaður
```

Figure 1: A snapshot from the Gold Standard.

The existence of the Gold Standard is fundamental to making this project possible and the standard is indeed created with the aim of being used for the training of ML algorithms in the Icelandic NLP field. However, in the case of this project, this wasn't a plug-and-play process, for a number of reasons:

1. The noun class has a label for proper nouns, but offers no further classification of the proper nouns.
2. Individual tokens are classified one by one and never as part of a bigger chunk, so only the first noun in a named entity noun phrase is labelled as a proper noun.

[5] *Tokens* are the separate words and punctuation that make up a text, and *tokenisation* is the process of splitting raw text into separate tokens. In this paper, *tokens* are generally used interchangeably with *words*.

[6] https://github.com/jonfd/nefnir

3. Foreign tokens are all labelled with the same tag, with no further distinction.

Apart from the above issues, a few errors were detected in the data and they will be contributed for improving the Gold Standard.

For the scope of this research, we took approximately the first 200,000 tokens from the Gold Standard, i.e. the first 20% of all the 13 different text types, to use as a training and testing corpus for our NER prototype.

3.3 Gazetteers

Since the earliest ideas of this project emerged, it has been clear that classifying proper nouns and foreign tokens in the corpus requires heavy work if done by hand only. To reduce this work, we gathered official records of person, organisation and place names as an input to an automatic pre-classification that afterwards would only require a manual review and correction. Here we will further describe how these gazetteers were gathered and processed.

Our most viable approach towards person names was using the Database of Modern Icelandic Inflection (Beygingarlýsing íslensks nútímamáls - BÍN)[7] (Bjarnadóttir 2005), a descriptive database of inflections in the Icelandic language. It contains around 15,000 given names and surnames of Icelanders. The database is available in whole for download in both *csv* and *sql* formats.

An even simpler interface to this data, that we benefitted from, is offered by Greynir (see Section 2.4). The source code for Greynir is accessible as the Python package *Reynir* (Vilhjálmur Þorsteinsson 2018). It includes the aforementioned BÍN data in compressed binary format, with a dedicated API. With only a simple wrapper on our side, the question whether a word is an Icelandic given name can be answered quite efficiently, considering the amount of underlying data.

Our sources for Icelandic places are the Icelandic Place Name Register (Örnefnaskrá) driven by the National Land Survey of Iceland (Landmælingar Íslands) and the Icelandic access address register (Staðfangaskrá), operated by Registers Iceland (Þjóðskrá).

Örnefnaskrá (Landmælingar Íslands 2018) is offered in two *GIS*[8] formats out of which *shapefile* was selected. It includes almost 120,000 records in total, divided into point, line and areal features. The names were fetched from the binary *dbf* database files with *dbfread*, a Python package. [9] When the different feature types had been merged and duplicates removed, approximately 67,000 names remained.

Staðfangaskrá (Þjóðskrá Íslands 2018) contains approximately 120,000 records of real estate information for Iceland. The data is available as a delimited text file and contains a lot of noise, requiring us to use regular expressions for filtering out the useful content. In the end, Staðfangaskrá contributed more than 11,000 additional place names to the project.

The final source utilised concerns organisations. The Directorate of Internal Revenue (Ríkisskattsstjóri) holds a record of all companies and individuals ever involved in a business of any kind in Iceland. The Company Registrar of Iceland (Fyrirtækjaskrá, Ríkisskattsstjóri 2018) is publicly open for limited queries on the web[10] but various IT companies offer bulk data dumps for a price. Our requests for the names of all companies in the file turned out to be too expensive to be viable for this project. With the help of a good friend it was possible to extract the names of 108,000 companies, registered since the year 1970, from the public web service. As expected, the file included some noise, and in the end 91,000 company names were used from the data.

3.4 Automatic pre-classification

To get us started in the classification, scripts were written in Python to apply the lists described earlier to the corpus. The implementation only uses packages included with the Python distribution, apart from the two described earlier, i.e. Reynir and dbfread. NLP libraries available in Python, like NLTK[11] that we had considered in the beginning, don't support Icelandic and thus were of limited use. If starting over again we would definitely use libraries like *Pandas* that we got to know along the way.

The code essentially makes a look-up in each available list, to categorise the proper nouns in the Gold Standard, and keeps track of the results. Ambiguities were registered to be resolved manually. This technique resembles the early NER gazetteer approaches, but can hardly be considered an NLP

[7] `http://bin.arnastofnun.is`
[8] *GIS*: Geographic Information System
[9] `https://dbfread.readthedocs.io`
[10] `https://www.rsk.is/fyrirtaekjaskra/`
[11] `http://www.nltk.org`

technique today. There were some twists, though, and a few of them follow.

In the case of person and company names, we implemented an iterative look-up to merge multi-token entities. For person names, measures were taken to include abbreviated non-listed middle names (in names such as "Guðni Th. Jóhannesson"). In addition, inflectional information on nouns in the Gold Standard was used to avoid merging adjacent names of two persons (e.g. "Í gær giftist María (nominative) Guðmundi (dative)").

Foreign phrases in the Gold Standard were too common to ignore in this work, and they included many named entities. As previously mentioned, all foreign tokens in the Gold Standard have the same tag, without any further classification, so methods needed to be developed to extract the named entities. A list of the relevant foreign phrases was prepared by considering sequential foreign tokens in the corpus. This huge list was reviewed manually to identify and classify the named entities. Finally, the resulting list was used as an input into the automatic classification.

Before entering the manual review phase, the output of the automatic classification was investigated for frequent double-, triple- and non-labelling of proper nouns. To battle the issue, we created an additional set of lists – with terms to include and to exclude for each category – for the automatic classification to incorporate.

It can be argued that these customised lists will make it harder to expand the training corpus later on. On the other hand, we now have the alternative of using the NER model itself for this job. This process might also be shortened somewhat by making better use of the annotation tool described in the following section, which simplifies the process of labelling named entities.

3.5 Manual review: BRAT

The pre-classification process described above proved useful in labelling most of the named entities found in the corpus, but inaccuracies were bound to appear. Some named entities went unlabelled, as the gazetteers are not (and can never be) exhaustive, some entities were labelled but miscategorised, some were false positives[12] and some labels included too many or too few words (incorrect boundaries).

The next step was therefore to review the whole training corpus by hand to clean up these inaccuracies. To make this process more efficient, we used *BRAT rapid annotation tool* (BRAT)[13] (Stenetorp et al. 2012). It is an online environment for text annotation that runs in a browser either driven by a production or a standalone web server. It is convenient for various annotation tasks, such as in NLP, and the annotated files are in a format readable by NeuroNER, the program we used for named entity recognition (see Section 3.7).

As BRAT accepts pre-annotated texts, we converted our pre-classified corpus to BRAT format, uploaded to a standalone server and completed the annotation in a browser. The BRAT format consists of two files for each text, a *txt* file and an *ann* file. The text file contains the original text and is never changed. The annotation file contains a list of all the named entities found in the text, marked with their position (character index) in the text file. This way, annotations can be added and removed at will without modifying the original corpus (Stenetorp et al. 2012).

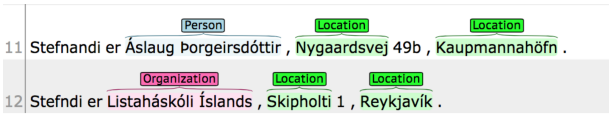

Figure 2: A snapshot from BRAT.

The ability to visualise the text and the annotations (see Fig. 2) made the review process much faster and easier than if it were to be done directly in text files. The tool also provided the added security of not mutating the corpus itself. In total, the task of manually reviewing the classification of 200,000 tokens in BRAT took about 25 hours.

3.6 ANNs and NER

As previously mentioned, the project described in this paper is twofold: 1) creating a labelled training corpus and 2) using that corpus to train an ANN for recognising and classifying named entities in written texts. The first task called for understanding of NLP and different NER methods, as well as general list manipulation and other programming techniques in Python. The latter task was much more research-oriented, as it involved getting familiar with the workings of ANNs and how they can be used for classification tasks such as NER.

[12] False positives are tokens that are labelled as belonging to some category (*positives*) but should not have been (*false*).

[13] `http://brat.nlplab.org/`

To set the scene for the second part of the project, the following sections are dedicated to the introduction of important concepts and ideas in ANNs. First we give an overview of the building blocks in ANNs, followed by a discussion of how RNNs and word embeddings are used to enhance the performance of named entity recognisers. Finally, we describe NeuroNER, the NER program used in this work, and how it makes use of these techniques.

3.6.1 ANNs

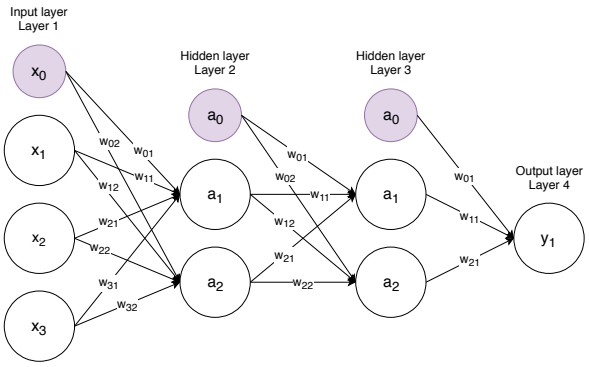

Figure 3: A simplified Artificial Neural Network.

An ANN (Fig. 3) is a state-of-the-art technique for many ML problems. Its roots can be traced back to the 1960s, but useful progress towards its application emerged decades later, in the 1980s (Jurafsky and Martin 2018).

An ANN is a computational framework that, by help of various algorithms, can learn very complex mappings from input values to outputs. In supervised learning[14], this is achieved by a labelled training set, i.e. a considerable amount of pre-labelled examples, giving true mappings from inputs to corresponding outputs. The process of making the network learn from existing examples is called "training the network". [15]

The training process can hardly be mentioned without also bringing two other related concepts to the table. Those are *validation* and *testing*. The labelled corpus is sectioned into those three sets with a typical division being 80%, 10% and 10% for training, validation and testing, respectively. As implied above, the training process has the objective of letting the network learn the mappings from inputs to outputs by adjusting the *weights* (sometimes called parameters) of the network (see below). The validation set is used to help with tuning some other parameters of the algorithms in question, these are referred to as hyperparameters. The validation set is also sometimes used to determine when to stop the training process, as too much training can be harmful. Finally, the test set is used only to evaluate the performance of the network. This separation between validation and test sets is important, to avoid tuning the system on the test set (Tjong Kim Sang and De Meulder 2003).

Building blocks of ANNs are loosely influenced by actual neurons in living organisms. ANNs are composed of a number of processing units (neurons), organised into layers. The first layer in the network is called the *input layer*, and the number of neurons in it corresponds to the number of input features (or variables) to the network. The last layer in the network, the *output layer*, has as many neurons as there are output values from the network. Between the input and output layers we have a variable number of so-called *hidden layers*. This naming convention derives from the fact that in supervised learning the values associated with the input and output layers are visible, but the others are not.

The number of hidden layers and the number of neurons in each layer, i.e. the depth and size of the network, varies according to the problem at hand. In conventional ANNs, there tended to be only one hidden layer, but as the technology developed and the computational power of graphics processing units was put to use, more hidden layers were often added, creating what is called a *deep* neural network. The depth here refers to the presence of more than one hidden layer.

As shown in Fig. 3, neurons in adjacent layers are interconnected to a high degree. Both the connections and the neurons have important roles when it comes to the calculations of the network. Each connection has a weight value w associated with it and brings an output, also called an activation a, from an upstream neuron to a downstream one as an input z, after being multiplied by w. Now, each node has many incoming connections so the total incoming value to neuron i in layer $n+1$ is a linear combination, where k denotes the

[14] Supervised learning is when the data includes output values that the network is trained to simulate, enabling it to predict others from new inputs. Unsupervised learning is when there are no pre-defined output values and the training has the objective to find and predict patterns or commonalities in the input data.

[15] This overview of ANNs is largely based on Chapter 7 in the book *Speech and Language Processing* (Jurafsky and Martin 2018).

number of neurons in layer n:

$$w_{0i} * a_0^n + w_{1i} * a_1^n + w_{2i} * a_2^n + ... + w_{ki} * a_k^n = z_i^{n+1}$$

A neuron, in all but the input layer of the network, takes its input value and applies a so called *activation function* to it, before it can be processed further, down the network. An activation function is a simple non-linear function. Examples of commonly used activation functions are $ReLU$ (rectified linear), $tanh$ and $sigmoid$. Without those, the network wouldn't be able to learn any non-linearities in the transformation of inputs to its output(s). Note the smaller grey circles in Fig. 3. These are not actual neurons, but *bias* terms that have a fixed value of 1, and contribute, as a_0, to the $w_{0i} * a_0^n$ term in the linear combination above.

As the description above demonstrates, the process of calculating outputs for given inputs to a weighted ANN is relatively simple. This process is referred to as *forward propagation*. The training process of an ANN has the objective of adjusting the values of the weights so that the overall error of the produced output from the network is minimised as compared to the actual labelled outputs. Formally, this is referred to as *minimising the cost function*. Determining how to adjust individual weights for overall error reduction is a complicated process. An algorithm called *back propagation* ultimately gives the partial derivatives of the cost function with respect to each weight in the network. Then the *gradient descent* algorithm can be applied to calculate updated weight values. To see how the error has reduced as a result of the new weights, forward propagation must be repeated. This leads to an iterative process, until no further cost reduction improvements are gained, whereby the training is said to converge (Jurafsky and Martin 2018).

The value we want an ANN to produce depends on the problem at hand. In NER applications we want the network to perform classification, i.e. for each token input it should tell us whether it is a named entity or not, and if so, in which category it belongs. In our 4-class NER setup, the network would need to produce a 5-fold classification, i.e. PER, LOC, ORG, MISC, and O[16]. This is done by using a suitable activation function in the neurons of the output layer. The activation function gives values between 0 and 1, that are interpreted as a probabilistic value. It gives the likelihood of the input word belonging to the category associated with the neuron in question. This way, for a network to provide a K-fold classification it needs K output neurons, except when there are only two possible groups; then a single neuron will suffice, as seen in Fig. 3.

[16] O (other) is a common way to indicate tokens that are not named entities.

3.6.2 RNNs

The tasks that ANNs perform, and the inputs they receive, vary greatly. In text-oriented tasks, such as NER, there is one fundamental feature of the input that needs to be considered. We are dealing with a temporal series of inputs of variable length, i.e. sentences, rather than individual words without a context. The order of the input words also matters, and preceding words can give clues as to how to classify words that appear later. The standard ANN, described earlier, has no functionality to meet such requirements, and therefore we need to update it to an RNN (see Fig. 4).

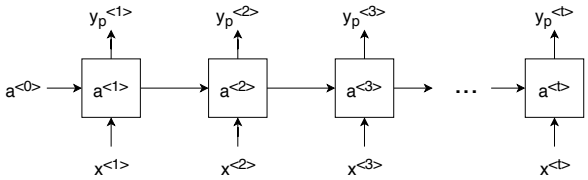

Figure 4: A Recurrent Neural Network.

The idea behind RNNs dates back to the 1980s (Elman 1990). In an iterative manner, RNNs process each input from a temporal series. The enhancement delivered by RNNs, as compared to standard ANNs, is the ability to bring state information between the hidden layers of the network as it moves forward in processing the input sequence. This is visualised in Fig. 4, where $x^{<t>}$ represents a single input in a sequence, like the t^{th} word in a sentence. $a^{<t>}$ is the activation of the hidden layer at time step t that is brought as an additional input to the hidden layer of the network for the subsequent time step. This way, the prediction of an output $y_p^{<t>}$ isn't just influenced by the corresponding input $x^{<t>}$, but also by the preceding inputs in the sequence.

RNNs come in a few different architectures for dealing with different sequence lengths of inputs vs. outputs. In NER, the output sequence and the input sequence are of equal length, as indicated in Fig. 4. This means that for each input value we get an output, i.e. the prediction of the class to which

the word in question belongs (Ng 2018).

Although RNNs are a fundamental update to ANNs for enabling the processing of sequences, like sentences in a natural language, they face an important difficulty, commonly referred to as *vanishing gradients*. Let's see an example:

My favourite **club**, ever since my family visited Spain many years back, is **Barcelona**.

My favourite **city**, ever since my family visited Spain many years back, is **Barcelona**.

The correct NER label for the last word in these example sentences (ORG and LOC, respectively) is determined by the third word in the sentence. In natural languages, dependencies like these can be very far apart. For a regular RNN, trained with an updated back-propagation algorithm called *back-propagation through time*, the gradients representing the error in the last time step have a hard time propagating back through so many time steps, to where the update of the weights to compensate for the error is needed (Ng 2018).

The solution integrated to the RNN is LSTM (Hochreiter and Schmidhuber 1997). Many variations of LSTM exist, but the main idea is to have a memory cell running through all time steps of the RNN. The memory cell stores a multidimensional bit vector that can stay intact, irrespective of the number of time steps travelled through. At each time step, a few gates control which proportion of the input to store in the memory cell, as well as what to forget from the memory of a previous time step. The content of the memory cell is integrated with the hidden state of an RNN unit, enhancing the long-term dependency capabilities of the network (Lample et al. 2016).

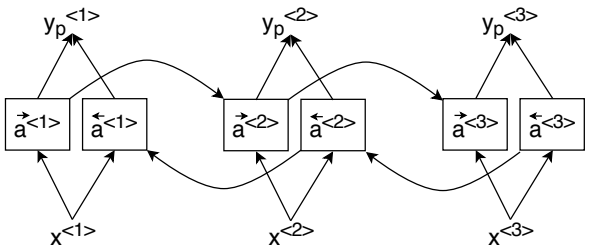

Figure 5: A Bidirectional RNN.

One additional step in these progressive updates of the standard ANN, used in many recent NER implementations, is that the LSTM RNN isn't unidirectional, as described earlier, but a bidirectional LSTM. As depicted in Fig. 5, one chain of RNNs stretches from left to right and another separate chain the other way around. These two chains have separate sets of trained weight values. This way, the information from both preceding and subsequent tokens in a sentence are available when a prediction is made for a particular token (Lample et al. 2016).

The methods discussed above, bidirectional LSTM RNNs, are the foundation of many NER systems that yield competitive results (Yadav and Bethard 2018). These are the methods used in NeuroNER, powering our Icelandic NER prototype. But before we can continue, we must mention one additional concept that can drastically enhance the results when extracting named entities from texts; namely word embeddings.

3.6.3 Word embeddings

In 2011, Collobert et al. presented a tool based on deep neural networks that combined many NLP tasks, including NER. Because of the tool's versatility, specifically engineered input features – a standard practice for tools with a narrow focus up until that time – were hard to implement. In this project, a new approach was used to input features into the ML NLP model. It marks the start of a new era in the field (Yadav and Bethard 2018), building on ideas and research stretching as far back as to mid-last century. Word embeddings, or distributional semantic models, have been quite a hot topic in recent years, as they have had a very positive influence on performance in many NLP tasks (Sahlgren 2015). Word embeddings are now the standard way to represent words in a meaningful way in ML NLP tasks.

Word embeddings are an example of an unsupervised learning method. In word embeddings, two ideas come together. First of all, the idea that the meaning of a word can be represented by an n-dimensional vector. Secondly, that this meaning can be captured by investigating other words that happen to stand close to the target word in texts, i.e., there is no need to study the target word itself (Jurafsky and Martin 2018).

To get the intuition of a vector representation of a word meaning, one can think of a set of words and try arranging them according to an arbitrary scale. As an easy example, let's start with some animals, arranged by size, from the smallest to the largest:

bee, earthworm, puffin, jellyfish, fox, eagle, shark, elephant, blue whale

If we add numeric values to indicate the size, these become one-dimensional vectors. It is easy to add another dimension, say the altitude of living space, and then we get the representation shown in Fig. 6.

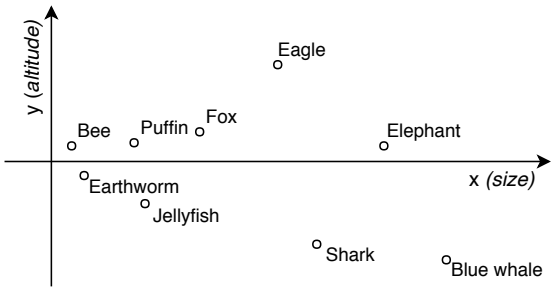

Figure 6: A simplified 2-D vector representation of a word meaning.

In fact, the possible number of scales a word's meaning could be quantified by is almost without limits. In word embeddings, the vector dimension, equivalent to the number of scales in the example above, usually ranges from 50 to 500 (Jurafsky and Martin 2018).

For the second idea above, the meaning of a word can be learnt by counting which other *informative words*[17] occur in its environment, defined by a *window size*[18]. Any two words having a similar set of nearby words have a higher probability of a closer meaning than others (Jurafsky and Martin 2018).

Embedding vectors, resulting from an approach like this, are large in dimension, corresponding to the size of the vocabulary in question. Each word gets a placeholder for its context count, but most of the values are zero, i.e. the vector is sparse, because most word pairs are never seen in the same context. Such vectors are inefficient as inputs to an ANN. Further advances are needed.

ANNs and other ML models are better off with dense vectors, that is, vectors with few zero elements, in moderate dimensions. Tokens are input to an ANN as *one-hot-vectors*, where all values are zero except for the bit corresponding to the word's index in the vocabulary. The first hidden layer of the network then contains as many neurons as there are dimensions in the word vectors. To translate a word to the z values that are input into the neurons of the first hidden layer, the embedding values are used as weights for the corresponding connections in the network.

One question still remains: How do we get the values for these embedding vectors? In NER, word embeddings and the corresponding weights can either be learned implicitly or by an external model. Implicit refers to the NER training process itself, after a random initialisation of the weights. The other, preferred way, is to obtain word vectors by a separate algorithm and initialise corresponding weight values accordingly in the NER model before training. Studies show that there is significant improvement in using pre-trained word embeddings, as compared to training them implicitly (Lample et al. 2016, Habibi et al. 2017, Lee, Dernoncourt, and Szolovits 2017). One reason for this is that external vectors can be created from unlabelled, huge data sets, while the labelled sets in NER training are bound to be much smaller. Thus, using larger data sets can improve the network's ability to know the meaning of a word not in its training set, or the meaning relative to another word seen while training. That way, the network can better generalise for a word it hasn't seen previously in its training set.

Word2Vec (Mikolov et al. 2013) is one of the available models to obtain word embeddings, and the one used in this project. It is language-agnostic, and can thus be used for Icelandic without modifications. It includes two algorithms, *skip-gram model with negative sampling (SGNS)* and *continuous bag-of-words* (CBOW). SGNS is a logistic regression binary classifier[19], trained to determine whether a word w_2 is likely to be within a defined context or window size to the current word w_1. The positive training examples used for this NLP model are the contexts from readily available running texts. Negative examples for the binary classifier are obtained by randomly sampling words from a lexicon.

In essence, this approach is a clever way of using the inherent word order of a corpus in one ML NLP problem as the labels that produce the valuable word embeddings used in another, harder problem (Jurafsky and Martin 2018).

[17] *Informative words* are words excluding the most common words in a language, such as *the*, *be*, *to* and *of* in English, that appear so frequently that they are of no use in discriminating contexts of other words.

[18] The window size determines how many words before and after the current word are to be studied.

[19] A logistic regression binary classifier can be thought of as a simplified version of an ANN, having no hidden layers, but making classifications just like a neuron classifier.

We pre-trained our own word embeddings from a large unlabelled corpus, to see if they would help improve our NER model. Using the Word2Vec architecture and an Icelandic tutorial on its use (Murphy 2014), we created word embeddings out of 543,354,729 tokens[20], taken from the Icelandic Gigaword Corpus (IGC) (Steinþór Steingrímsson et al. 2018).[21] The parameters for Word2Vec were set to *dimension*=200, *window size*=5 and *min count*=4. The obtained results are presented in Section 4.

3.7 NeuroNER

When searching for open-sourced ML NER architectures that use ANNs, quite early on we came across NeuroNER[22]. This is the system we ended up using for our NER prototype.

NeuroNER uses deep learning methods to recognise and classify named entities. It can train new NER language models from scratch, and also offers some trained language models that can be used to perform NER on English texts. To make the annotation process smoother and provide a visualisation of the output, NeuroNER interfaces with the BRAT environment described in Section 3.5. The program accepts annotated corpus files in either the BRAT format or the format from the CoNLL-2003 shared task (Tjong Kim Sang and De Meulder 2003). Tensorflow[23], a numerical computation library, is used for implementing NeuroNER. The training process can be monitored in real time and viewed in TensorBoard, a web application suite for viewing and understanding TensorFlow data. Plots are also generated throughout the whole training process and show the evolution of the score metrics, confusion matrices for the different labels, and classification reports.

Once a model has been trained, it can be used for recognising named entities in texts it has not seen previously.

3.7.1 How NeuroNER fits the project

Below we will list the advantages and disadvantages of NeuroNER in regards to this project.

NeuroNER delivers state-of-the-art results:
NeuroNER has been reported to reach F_1 of 90.5% on the English CoNLL-2003 data set (*PER*, *LOC*, *ORG*, *MISC*), and 97.7% on the i2b2 2014 data set.[24] These results are on par with state-of-the-art performance at the time of writing (Dernoncourt, Lee, and Szolovits 2017).

NeuroNER is published scientific work:
Papers have been published describing the properties of the program (Dernoncourt, Lee, and Szolovits 2017), on how it was used for removing personal information from patient notes and details regarding its architecture (Dernoncourt, Lee, Uzuner, et al. 2016), and on how so-called transfer learning can prove useful in training ANN models (Lee, Dernoncourt, and Szolovits 2017).

NeuroNER is easy to use:
NeuroNER's authors describe it as the first ANN-based NER system for non-experts. When training a new language model, only three directories must be provided, each containing corpus data for different parts of the process: training, validation, and testing. All necessary parameters are specified, with default values, in a configuration file the user can change. Therefore, there is no need to examine the code in order to run the program. In fact, the only thing the user needs to change in the configuration file is to specify the path to the data set.

Our experience of ANNs is limited, and because of NeuroNER's usability, we could focus on getting the experiment running before diving into the particulars. Additionally, the integration between BRAT and NeuroNER was a convenient way to conduct the manual review and inject the resulting files directly into NeuroNER.

NeuroNER has one disadvantage:
NeuroNER is mostly language independent, the only exception being the method for tokenising the input text. The way sentences are split up into individual tokens varies somewhat between languages. NeuroNER includes two third-party tokenisers to choose from, and neither of them has support for Icelandic. Our corpus, the Gold Standard, is already tokenised, but to make the format acceptable to NeuroNER we had to implement one new function in the source.

Looking back, we are satisfied with our choice, as this easy-to-use solution became our stepping stone into a very interesting yet complex field of study.

[20] `http://www.malfong.is/files/rmh_textaflokkar_is.pdf`

[21] This part of IGC contains mostly official documents.

[22] `http://neuroner.com/`

[23] `http://www.tensorflow.org/`

[24] The performance difference for the two data sets stems from the fact that the latter one is directed towards the problem of de-identification of patient notes, a quite specific task.

3.7.2 NeuroNER architecture

NeuroNER incorporates the methods and techniques discussed in the previous sections. It uses the LSTM variant of bidirectional RNNs, described in Section 3.6.2, and is an example of an architecture with a combined input representation, i.e. both word and character embeddings.

Taking a further look at the system, we can divide it into three layers:

- Character-enhanced token embedding layer
- Label prediction layer
- Label sequence optimisation layer

The character-enhanced token embedding layer maps each token into a vector representation, using two types of embeddings: a token embedding and a character-level token embedding. The token embedding is the one we refer to as a word embedding in Section 3.6.3, and it is used to capture the meaning of a word as a vector.

Word embeddings do a great job of capturing semantics, but they fall short when dealing with unknown words, misspellings and other variations in the lexicon (Dernoncourt, Lee, Uzuner, et al. 2016). To remedy this, character-level token embeddings are implemented. They are trained implicitly, i.e. by the NER model itself. For the character-level token embeddings, instead of looking at each token in a sequence of tokens, we look at each character within a token. Each character of a token is mapped to a vector, which is fed into a bidirectional LSTM, giving just one output for each network direction and token. As depicted in Fig. 7, these activation vectors, i.e. the output, are appended to the token embeddings, giving a word representation to be fed into the next layer of NeuroNER.

The sequence of vectors containing the word representations is the input to the second layer, the label prediction layer. This is the layer that calculates the probability of a label (such as PER, LOC, ORG) for each individual token. This layer contains a bidirectional LSTM, similar to the character embedding layer, plus a single-layer feed-forward ANN which returns the probability vector for each token.

Once the probability vectors have been determined, there must be a way to determine the correct label for a token. This is accomplished in the label sequence optimisation layer. It could be done by always choosing the label with the highest probability for each token, but in that case, the dependencies between labels that tend to stand close together are partly lost. The solution the authors used is adding transition probabilities to the probability of a particular label. The transition probabilities are calculated as $T[i, j]$, i.e., how likely is it that a token with the label i is followed by a token with the label j (Dernoncourt, Lee, Uzuner, et al. 2016). An easily understandable example of this could be the token sequence "Háskólinn í Reykjavík". In the probability vector from the prediction layer, the probability of "Reykjavík" being a LOC could well be higher than the probability of it being a part of an ORG. But because of the optimisation layer the recogniser will do this right because the transition probabilities for In the probability vector from the prediction layer, the probability of "Reykjavík" being a LOC could well be higher than the probability of it being a part of an "Reykjavík" as an ORG following the other two tokens will be high.

Label	*Count*	*%*
PER	3045	40.4
LOC	1748	23.2
ORG	1768	23.4
MISC	977	13.0
Total	**7538**	

Table 1: Number of named entities in the 200,000 token training corpus.

4 Experiments and results

Before presenting the results, let's quickly review the setup of the experiment.

The data set used as a training corpus contained in total approximately 200,000 tokens, with 7,538 named entities. The split between entity types is quite imbalanced, with the PER category being the largest one and the MISC category by far the smallest (see ratio in Table 1).

The data was arranged into two sets of different sizes, 100,000 and 200,000 tokens, each split into training (80%), validation (10%) and test (10%) sets. Four different configurations were tested, on both sizes and for both implicitly and externally trained word-embeddings (see Section 3.6.3).

All the parameters in the NeuroNER's configuration file, controlling the structure of the model, along with the hyperparameters directed

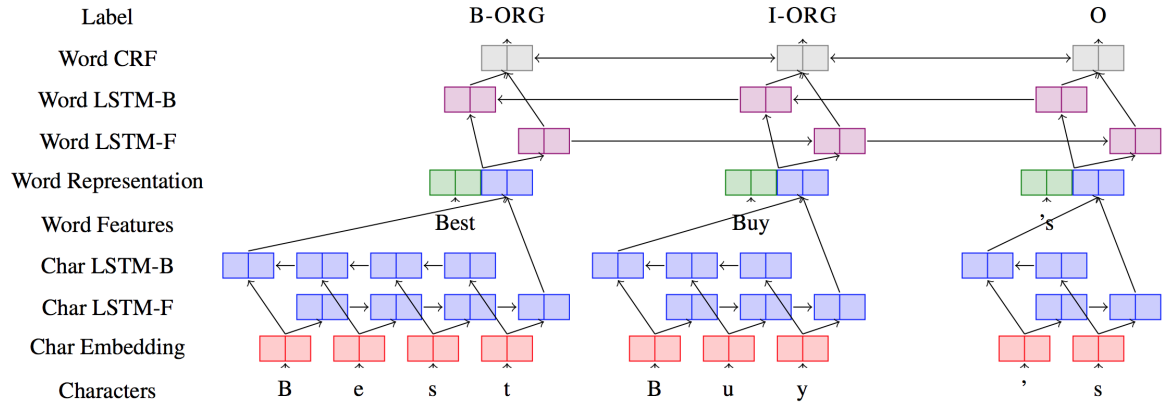

Figure 7: Word + character level NN architecture for NER (from Yadav and Bethard 2018, p. 2151).

W. embeddings	Implicit		External	
Corpus size	100K	200K	100K	200K
PER	71.8	76.1	95.2	93.3
LOC	61.8	65.6	81.8	85.6
ORG	23.5	40.5	62.7	69.2
MISC	3.2	28.3	14.8	41.5
Overall	**55.5**	**61.8**	**80.6**	**81.3**

Table 2: F_1 score (%) of different training configurations.

towards the learning process, were left at their default values. The only exception to this is the *token_embedding_dimension* parameter, controlling the length of the word vectors i.e. the number of neurons the words are mapped into when input to the network. This value was increased from 100 to 200 for the external word embeddings.

In the training, early stop was applied by default when no improvement had been seen on the validation set for ten consecutive epochs. The language model used is based on the network weights taken from the epoch where F_1 last peaked for the validation set.

Evaluation was done automatically by NeuroNER according to CoNLL practices.

Table 2 presents the F_1 score for the models of all training configurations, i.e. for the two training corpus sizes, with implicit and external word embeddings. Further details on the underlying precision and recall values can be seen in Fig. 8.

These figures indicate a benefit of using externally trained word embeddings, with an overall F_1 score of 81.3%. The results will be discussed further in the following section.

5 Discussion

The results observed in Section 4 are encouraging, given the rather small corpus. Particularly noteworthy are the improved results when incorporating pre-trained word embeddings.

In the following discussion, the configuration (model) we refer to by default is the one showing the highest F_1, i.e. the one trained on the larger corpus with pre-trained word embeddings. Where applicable, we will use the English and German CoNLL data set for comparison (Tjong Kim Sang and De Meulder 2003).

Initially, one of the goals of this work was to study the influence on the performance of the NER prototype when doubling the training corpus. In the results without the pre-trained word embeddings, we observe a considerable improvement on F_1 when doubling the corpus size, from 55.5% to 61.8%. This was expected, as the training set in the 100,000 token corpus only contains around 80,000 tokens, and a very limited number of named entity examples to learn from. For an even larger corpus, we would expect this trend to continue, as indicated in Fig. 9. However, a more efficient approach to enhancing the performance proved to be incorporating pre-trained word embeddings. In our case this improves F_1 from 61.8% to 81.3%. This is an interesting outcome that deserves a closer look.

The sources we have studied agree that pre-trained word embeddings are beneficial for NER tasks (e.g. Y. Wu et al. 2015; Dernoncourt, Lee, and Szolovits 2017; Demir and Özgür 2014). However, most don't report more than a few percent increase in F_1 by introducing pre-trained

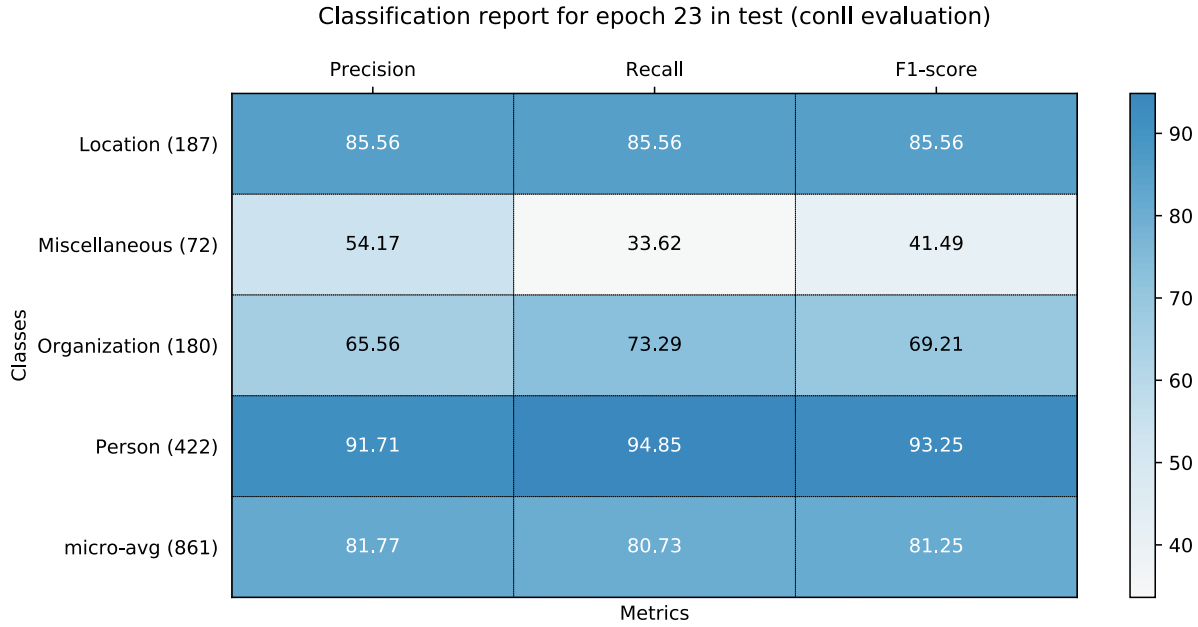

Figure 8: Classification report for the test set (200,000 tokens with external word embeddings).

word embeddings. One research that stands out was conducted by Lample et al. (2016). They have an architecture similar, if not identical, to that of NeuroNER, and report the influence of each model component on F_1. Out of features like CRF, character-level representation and dropout, pre-trained word embeddings gave by far the largest boost to the performance, 7.3% on the English CoNLL-2003 set (the other components adding below 1.8% to F_1).

Intuitively, we deduce that the main reason we are experiencing this huge influence of pre-trained word embeddings on F_1, is the small size of our corpus. For a small training set, the model will more often encounter unseen words in the test set. In our data, 60% of the incorrectly labelled words had not been seen in the training set. When the large collection of word embeddings is added to the pool, the chances that a word is known increase enormously. Add to that the contextual information that the pre-trained word embeddings introduce, and the performance is bound to improve considerably.

With a larger corpus, say by doubling it once again, we believe, from the trend, that F_1 without external embeddings might end up between the earlier score and the one seen currently with pre-trained word embeddings. At the same time the results with pre-trained embeddings indicate a much slower growth in F_1 by growing the corpus. Thus the gain of adding external embeddings will slow down with a larger training corpus and we would see gains similar to what others have reported (see Fig. 9). The gain in F_1 of adding pre-trained word embeddings can be visualised as the vertical distance between the the lines in the graph. The dotted portion indicates our prediction of a possible result for a larger training corpus.

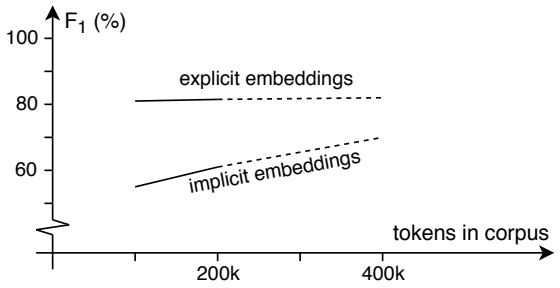

Figure 9: Extrapolating F_1 for a larger corpus.

Another related reason as to why word embeddings from a large external corpus work so well for our model may lie in the structure of the language. Icelandic, a morphologically rich inflected language, presents special challenges for various NLP tasks, such as NER. Nouns, generally the building blocks of named entities, have up to 16 unique inflectional forms, and verbs and adjectives additionally have over a hundred different forms. This greatly increases the vocabulary size of a corpus, and causes a problem with data sparsity, as

Demir and Özgür point out in the case of Turkish and Czech (2014). The implication is that a NER system may not recognise a named entity in the test set if it has seen it in a different form in the training set.[25] We could try lemmatising the tokens in a corpus and use the normalised output for NER, but then we would lose important contextual information about the named entities and their neighbours. The pre-trained word embeddings contribute many examples to the model of different word forms that don't appear in the training set, and the likelihood of correctly labelling them increases as a result.

We observe a considerable difference in the performance of different categories. Especially promising are the results in the PER category, with F_1 of 93.3%. Recall is high, 94.85% (see Fig. 8), which means that almost all person names in the test set were found. Several factors may explain the top performance in this category. Most importantly, person names are by far the most frequent entity type in the training corpus, almost double that of the LOC and ORG categories. Person names are often constructed in a similar manner, with Icelandic full names usually composed of one or two given names and a surname ending in "-son" or "-dóttir". Furthermore, they are also almost always capitalised, and they are not unique (many people can have the same name), so each person name is bound to appear more often than each organisation name. All this adds up to help the system label these entities as persons.

The system also performs quite well on the LOC category. It is the nature of a corpus sampled from any geographic area that some locations appear more often than others, in this case "Ísland" and "Reykjavík", to name two of the most common. That means the system is much more likely to know them and thus gets them right every time. Another property of place names is that they tend to be single word entities, and are capitalised, with a few exceptions. As a result, detecting word boundaries becomes less of a challenge.

In contrast, in the ORG category, word boundaries are a problem, as organisations are often composed of more than one word, not necessarily capitalised (e.g. "Samband lífeyrisþega ríkis og bæja"). LOC and ORG labels are equally common in the corpus, but ORG performs significantly worse. Further properties of organisation names may explain this difference. Organisation names don't necessarily follow a set structure as person names do, so they may be harder to recognise, and sometimes it can be hard to decide whether an entity is an organisation name or a product. This may cause overlap with the MISC category.

The MISC category, incidentally, was the most problematic one, with F_1 of 41.49%. Recall is particularly low (33,62%), meaning many MISC entities are not found, and precision is not particularly high either (54.17%), so many entities are mislabelled.

The confusion matrix from the classification of the test set (Fig. 10) shows the performance of the classifier, displaying how many of the tokens in the test set were correctly labelled (the diagonal line running from the top left corner to the bottom right), and where mislabelling occurs. The MISC category contains the most outliers, with a total of 115 entities missed out of 202 in total. We can compare this performance to that of the PER category, where only 20 entities out of 588 are mislabelled.

There are only around 1000 MISC entities in the whole corpus, and a lot of variation in how they are constructed, which makes detecting them harder, even for human annotators. Some are long book or movie titles, some are complicated product names with numbers and hyphens, and there is no correlation within the category. With a substantially larger corpus, and a further division within this category, introducing categories such as products, artworks, and events, we might see an improvement.

Of note is the negligible improvement from doubling the corpus, when using external word embeddings. This makes us wonder whether we are approaching the upper limit, performance-wise. Considering CoNLL F_1 figures for other morphologically complex languages, this is a reasonable deduction. NER systems for German and Turkish report numbers similar to ours (Demir and Özgür 2014, Yadav and Bethard 2018).

Our small training set has been brought up several times in this discussion. It follows that the test set contains only around 20,000 tokens and 870 named entities. This means that our results may be affected by anomalies in the test set, not representative of normal text. This must be considered

[25] For example, we can imagine that the person name "Egill" (nominative) is tagged as such in the training set and then appears as "Agli" (dative) or "Egils" (genitive) in the test set, and is mislabelled as a result.

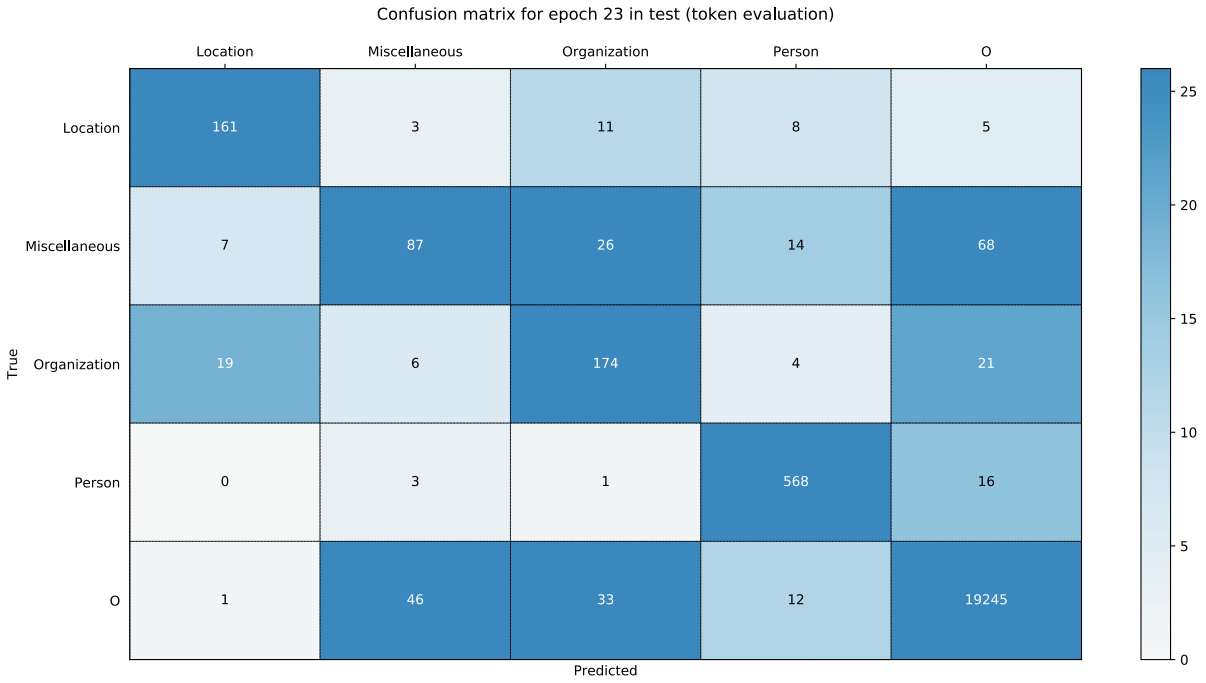

Figure 10: Confusion matrix for the classification of the test set (200,000 tokens with external word embeddings).

when looking at the results.

Forman and Cohen (2004) have shown that for some ML tasks, such as classification tasks with limited data, the deciding factor in how well the model performs is the number of positive examples in the data, not the number of negative examples. In the case of NER, the positive examples are the labelled entities in the corpus, and the negative examples are all other tokens – a great majority. This implies that increasing the concentration of labelled named entities in the training corpus is more important than increasing the total size of the corpus.

In constructing our corpus, we were aiming for a data set representative of written Icelandic texts, and therefore we included all sentences, regardless of whether they contained named entities or not. The CoNLL data set (Tjong Kim Sang and De Meulder 2003) for English and German has a much higher concentration of named entities than our corpus, as seen in Fig. 12. The total size of the English CoNLL-2003 data set is 301,000 tokens in 22,000 sentences, where one third is devoted to validation and test sets. The total number of named entities in the set is around 35,000. As our corpus contains 200,000 tokens, this data set is only larger by a third, but it contains five times as many named entities. This is an indication that there is much room for improvement in our training corpus.

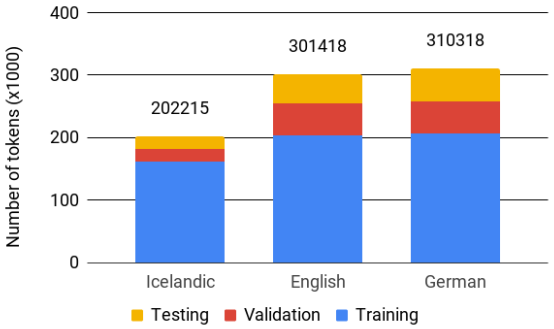

Figure 11: Size of the Icelandic corpus and the English and German CoNLL corpora.

Assuming a similar ratio of named entities in our corpus and the rest of the Gold Standard, we conclude that tagging the whole Gold Standard would result in roughly the same number of named entities as in the English CoNLL data set; around 35,000 in total. Such a corpus would be a valuable asset for further research on NER in Icelandic, and could serve as a benchmark corpus for evaluation of future NER systems.

Code and procedures exist for classifying the remaining 80% of the Gold Standard, but a more effective approach may be using the NER prototype

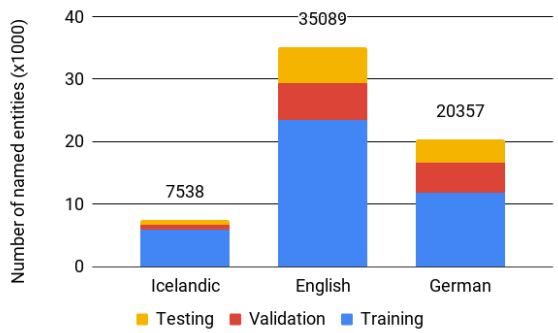

Figure 12: Named entity count in the Icelandic corpus and the English and German CoNLL corpora.

itself for classification. Whichever method is chosen, the bulk of the work will always be in reviewing the output manually. This would be an obvious next step in the research, were it to be continued.

Another improvement upon the prototype would be to introduce a gazetteer lookup, in order to catch named entities the model may have missed. This is a common method for increasing F_1 in ML NER systems. For instance, it is used in half of the systems reviewed in Yadav's and Bethard's recent survey of NER systems (2018). As we have already gathered gazetteers containing about 185,000 named entities, in the PER, LOC and ORG categories, this would be a simple way to try and improve the performance of the named entity recogniser.

6 Conclusion

Our named entity recogniser shows that, even with a small training corpus, it can serve to extract most of the named entities appearing in written Icelandic. This is especially true when using external word embeddings trained on a larger corpus. We consider continuing this work, and turning the prototype into a usable tool, a worthy goal.

Apart from the prototype itself, there is value in having a scalable training corpus for NER tasks in Icelandic. Hopefully, this corpus can serve as a resource for other researchers in the field of NLP.

Acknowledgements

We would like to extend our thanks to Hrafn Loftsson for his guidance, helpful suggestions, and insight throughout this project.

We also want to thank Vilhjálmur Þorsteinsson, Steinþór Steingrímsson, and Ólafur Sindri Ólafsson for their advice and input, and Páll Hilmarsson for his valuable help.

References

Agerri, Rodrigo and German Rigau (2017). "Robust Multilingual Named Entity Recognition with Shallow Semi-Supervised Features". In: *CoRR* abs/1701.09123. arXiv: 1701.09123.

Ahmed, Imran and Sathyraj R (2015). "Named Entity Recognition by Using Maximum Entropy". In: *International Journal of Database Application and Theory* 8.

Bjarnadóttir, Kristín (2005). "Modern Icelandic Inflections". In: *Nordisk Sprog-teknologi. Årbog 2005*. Ed. by Henrik Holmboe. Copenhagen: Museum Tusculanums Press, University of Copenhagen.

Chiticariu, Laura et al. (2010). "Domain Adaptation of Rule-Based Annotators for Named-Entity Recognition Tasks". In: *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*. Cambridge, MA: Association for Computational Linguistics, pp. 1002–1012.

Chiu, Jason P. C. and Eric Nichols (2015). "Named Entity Recognition with Bidirectional LSTM-CNNs". In: *CoRR* abs/1511.08308. arXiv: 1511.08308.

Chowdhury, G. (2003). "Natural language processing". In: *Annual Review of Information Science and Technology* 37, pp. 51–89.

Collins, Michael and Yoram Singer (1999). "Unsupervised Models for Named Entity Classification". In: *In Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, pp. 100–110.

Collobert, Ronan and Jason Weston (2008). "A Unified Architecture for Natural Language Processing: Deep Neural Networks with Multitask Learning". In: *Proceedings of the 25th International Conference on Machine Learning*. ICML '08. Helsinki, Finland: ACM, pp. 160–167. ISBN: 978-1-60558-205-4. DOI: 10.1145/1390156.1390177.

Daðason, Jón Friðrik (2018). *Nefnir [Computer software]*. URL: https://github.com/jonfd/nefnir.

Demir, H. and A. Özgür (2014). "Improving Named Entity Recognition for Morphologically Rich Languages Using Word Embeddings". In:

2014 13th International Conference on Machine Learning and Applications, pp. 117–122. DOI: 10.1109/ICMLA.2014.24.

Derczynski, Leon et al. (2017). "Results of the WNUT2017 Shared Task on Novel and Emerging Entity Recognition". In: *Proceedings of the 3rd Workshop on Noisy User-generated Text*. Copenhagen, Denmark: Association for Computational Linguistics, pp. 140–147. DOI: 10.18653/v1/W17-4418.

Dernoncourt, Franck, Ji Young Lee, and Peter Szolovits (2017). "NeuroNER: an easy-to-use program for named-entity recognition based on neural networks". In: *Conference on Empirical Methods on Natural Language Processing (EMNLP)*.

Dernoncourt, Franck, Ji Young Lee, Özlem Uzuner, et al. (2016). "De-identification of Patient Notes with Recurrent Neural Networks". In: *CoRR* abs/1606.03475. arXiv: 1606.03475.

Elman, Jeffrey L (1990). "Finding structure in time". In: *Cognitive science* 14.2, pp. 179–211.

Ferro, L. et al. (2005). *TIDES 2005 Standard for the Annotation of Temporal Expressions*. Tech. rep. MITRE.

Finkel, Jenny Rose, Trond Grenager, and Christopher Manning (2005). "Incorporating Non-local Information into Information Extraction Systems by Gibbs Sampling". In: *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*. ACL '05. Ann Arbor, Michigan: Association for Computational Linguistics, pp. 363–370. DOI: 10.3115/1219840.1219885.

Forman, George and Ira Cohen (2004). "Learning from Little: Comparison of Classifiers Given Little Training". In: *Knowledge Discovery in Databases: PKDD 2004*. Ed. by Jean-François Boulicaut et al. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 161–172. ISBN: 978-3-540-30116-5.

Grishman, Ralph and Beth Sundheim (1996). "Message Understanding Conference-6: A Brief History". In: *Proceedings of the 16th Conference on Computational Linguistics - Volume 1*. COLING '96. Copenhagen, Denmark: Association for Computational Linguistics, pp. 466–471. DOI: 10.3115/992628.992709.

Habibi, Maryam et al. (2017). "Deep learning with word embeddings improves biomedical named entity recognition". In: *Bioinformatics* 33.14, pp. i37–i48.

Helgadóttir, Sigrún, Hrafn Loftsson, and Eiríkur Rögnvaldsson (2014). "Correcting Errors in a New Gold Standard for Tagging Icelandic Text". In: *Proceedings of the Ninth International Conference on Language Resources and Evaluation, LREC 2014, Reykjavik, Iceland, May 26-31, 2014*. Pp. 2944–2948.

Hochreiter, Sepp and Jürgen Schmidhuber (1997). "Long short-term memory". In: *Neural computation* 9.8, pp. 1735–1780.

Huang, Zhiheng, Wei Xu, and Kai Yu (2015). "Bidirectional LSTM-CRF Models for Sequence Tagging". In: *CoRR* abs/1508.01991. arXiv: 1508.01991.

Íslensk málnefnd (2016). *Ritreglur*. Íslensk málnefnd. URL: http://islenskan.is/images/ritreglur-IM-2016.pdf.

Jurafsky, Daniel and James H. Martin (2018). *Speech and Language Processing*. URL: https://web.stanford.edu/~jurafsky/slp3/ed3book.pdf (visited on 11/23/2018).

Kokkinakis, Dimitrios et al. (2014). "HFST-SweNER. A New NER Resource for Swedish". In: *Proceedings of the 9th edition of the Language Resources and Evaluation Conference (LREC), Reykjavik 26 - 31 May 2014*. Pp. 2537–2543. ISBN: 978-2-9517408-8-4.

Lample, Guillaume et al. (2016). "Neural architectures for named entity recognition". In: *CoRR* abs/1603.01360. arXiv: 1603.01360.

Landmælingar Íslands (2018). *Örnefnaskrá*. URL: http://atlas.lmi.is/LmiData/index.php (visited on 08/27/2018).

Lee, Ji Young, Franck Dernoncourt, and Peter Szolovits (2017). "Transfer Learning for Named-Entity Recognition with Neural Networks". In: *CoRR* abs/1705.06273. arXiv: 1705.06273.

Loftsson, Hrafn and Eiríkur Rögnvaldsson (2007). "IceNLP: a natural language processing toolkit for icelandic". In: *INTERSPEECH 2007, 8th Annual Conference of the International Speech Communication Association, Antwerp, Belgium, August 27-31, 2007*, pp. 1533–1536.

Loftsson, Hrafn, Jökull H. Yngvason, et al. (2010). "Developing a PoS-tagged corpus using existing tools". In: *7th SaLTMiL Workshop on Creation and Use of Basic Lexical Resources for*

Less-Resourced Languages. LREC 2010. Ed. by Francis M. Tyers og Mikel L. Forcada Sarasola Kepa, pp. 53–60.

Manning, Christopher D., Prabhakar Raghavan, and Hinrich Schütze (2008). *Introduction to Information Retrieval*. New York, NY, USA: Cambridge University Press.

Mikolov, Tomas et al. (2013). "Efficient Estimation of Word Representations in Vector Space". In: *CoRR* abs/1301.3781. arXiv: 1301.3781.

Mohit, Behrang (2014). "Named Entity Recognition". In: *Natural Language Processing of Semitic Languages*. Ed. by Imed Zitouni. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 221–245.

Murphy, Alex (2014). *Orð2Vec [Tutorial]*.

Nadeau, David and Satoshi Sekine (2007). "A Survey of Named Entity Recognition and Classification". In: *Lingvisticae Investigationes* 30. DOI: 10.1075/li.30.1.03nad.

Nadeau, David, Peter D. Turney, and Stan Matwin (2006). "Unsupervised Named-Entity Recognition: Generating Gazetteers and Resolving Ambiguity". In: *Advances in Artificial Intelligence*. Ed. by Luc Lamontagne and Mario Marchand. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 266–277. ISBN: 978-3-540-34630-2.

Ng, Andrew (2018). *Recurrent Neural Networks | Sequence Models*. URL: https://www.youtube.com/watch?v=efWlOCE_6HY&list=PL1w8k37X_6L_s4ncq-swTBvKDWnRSrinI&index=1 (visited on 11/15/2018).

Nikulásdóttir, A. B., J. Guðnason, and S. Steingrímsson (2017). *Language Technology for Icelandic 2018-2022. project plan*. Ministry of Education, Science and Culture.

Paliouras, Georgios et al. (2000). "Learning Decision Trees for Named-Entity Recognition and Classification". In: *Proceedings of the ECAI Workshop on Machine Learning for Information Extraction.*

Passos, Alexandre, Vineet Kumar, and Andrew McCallum (2014). "Lexicon Infused Phrase Embeddings for Named Entity Resolution". In: *CoRR* abs/1404.5367. arXiv: 1404.5367.

Ramshaw, L. A. and M. P. Marcus (1995). "Text Chunking Using Transformation-Based Learning". In: *Natural Language Processing Using Very Large Corpora*. Ed. by Susan Armstrong et al. Dordrecht: Springer Netherlands, pp. 157–176. ISBN: 978-94-017-2390-9. DOI: 10.1007/978-94-017-2390-9_10.

Rau, L. F. (1991). "Extracting company names from text". In: *[1991] Proceedings. The Seventh IEEE Conference on Artificial Intelligence Application*. Vol. i, pp. 29–32. DOI: 10.1109/CAIA.1991.120841.

Ríkisskattsstjóri (2018). *Fyrirtækjaskrá*. URL: https://www.rsk.is/fyrirtaekjaskra/ (visited on 09/21/2018).

Riloff, Ellen and William Phillips (2004). "An Introduction to the Sundance and AutoSlog Systems". In: *University of Utah School of Computing*.

Ritter, Alan et al. (2011). "Named Entity Recognition in Tweets: An Experimental Study". In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. EMNLP '11. Edinburgh, United Kingdom: Association for Computational Linguistics, pp. 1524–1534. ISBN: 978-1-937284-11-4.

Sahlgren, Magnus (2015). *A brief history of word embeddings (and some clarifications)*. URL: https://www.linkedin.com/pulse/brief-history-word-embeddings-some-clarifications-magnus-sahlgren/ (visited on 11/23/2018).

Seif, George (2018). *Deep Learning vs Classical Machine Learning*. URL: https://towardsdatascience.com/deep-learning-vs-classical-machine-learning-9a42c6d48aa (visited on 12/12/2018).

Steingrímsson, Steinþór et al. (2018). "Risamálheild: A Very Large Icelandic Text Corpus". In: *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*. Ed. by Nicoletta Calzolari (Conference chair) et al. Miyazaki, Japan: European Language Resources Association (ELRA). ISBN: 979-10-95546-00-9.

Stenetorp, Pontus et al. (2012). "BRAT: A Web-based Tool for NLP-assisted Text Annotation". In: *Proceedings of the Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics*. EACL '12. Avignon, France: Association for Computational Linguistics, pp. 102–107.

Tanabe, Lorraine et al. (2005). "GENETAG: a tagged corpus for gene/protein named entity

recognition". In: *BMC Bioinformatics* 6.1, S3. ISSN: 1471-2105. DOI: 10.1186/1471-2105-6-S1-S3.

Tjong Kim Sang, Erik F. (2002). "Introduction to the CoNLL-2002 Shared Task: Language-Independent Named Entity Recognition". In: *Proceedings of CoNLL-2002*. Taipei, Taiwan, pp. 155–158.

Tjong Kim Sang, Erik F. and Fien De Meulder (2003). "Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition". In: *Proceedings of CoNLL-2003*. Ed. by Walter Daelemans and Miles Osborne. Edmonton, Canada, pp. 142–147.

Tryggvason, Aðalsteinn (2009). *Named Entity Recognition for Icelandic. Research report*. Reykjavík University.

Vilhjálmur Þorsteinsson, Miðeind ehf. (2018). *Welcome to Reynir — Reynir 1.3.0 documentation*. URL: https://greynir.is/doc/ (visited on 08/27/2018).

Wu, Yu-Chieh et al. (2006). "Extracting Named Entities Using Support Vector Machines". In: *Knowledge Discovery in Life Science Literature*. Ed. by Eric G. Bremer et al. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 91–103. ISBN: 978-3-540-32810-0.

Wu, Yonghui et al. (2015). "A Study of Neural Word Embeddings for Named Entity Recognition in Clinical Text". In: *AMIA, Annual Symposium proceedings. AMIA Symposium* 2015, pp. 1326–33.

Yadav, Vikas and Steven Bethard (2018). "A Survey on Recent Advances in Named Entity Recognition from Deep Learning models". In: *Proceedings of the 27th International Conference on Computational Linguistics*, pp. 2145–2158.

Zhang, Shaodian and Noémie Elhadad (2013). "Unsupervised biomedical named entity recognition: Experiments with clinical and biological texts". In: *Journal of biomedical informatics* 46 6, pp. 1088–98.

Þjóðskrá Íslands (2018). *Staðfangaskrá*. URL: https://www.skra.is/einstaklingar/gagnagrusk/nidurhal/ (visited on 08/30/2018).