Title (EN): Domain-Specific NER Adaptation

In the past years, the Named Entity Recognition (NER) technology has been under an active development and enjoy a significant increase in popularity and usage in the academic and industrial sphere. Nevertheless, vast majority of the developed NER systems have been developed as general-purpose systems. While they can perform well on multiple domains (macro level), on specific domains (micro level) their performance quality might be low. The ultimate goal of the thesis is to develop domain-specific NER models. Guidelines:

- Get familiar with the NER technology and available NER frameworks.
- Investigate possible datasets for domain-specific training of NER.
- Develop NER training datasets for several selected domains (e.g. sports, politics, music, etc.).
- Train a domain-specific NER model using existing frameworks, such as DBpedia Spotlight or StanfordNER.
- Validate and evaluate the developed domain-specific NER models.

FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Master's thesis

Domain-specific Named Entity Recognition

Bc. Bogoljub Jakovcheski

Department of software engineering
Supervisor: Ing. Milan Dojčinovski

June 27, 2018

Acknowledgements

I would like to thank my family and friends and especially to my supervisor Mr. Ing. Dojčinovski for support during writing this thesis.

Declaration

I hereby declare that the presented thesis is my own work and that I have cited all sources of information in accordance with the Guideline for adhering to ethical principles when elaborating an academic final thesis.

I acknowledge that my thesis is subject to the rights and obligations stipulated by the Act No. 121/2000 Coll., the Copyright Act, as amended. In accordance with Article 46(6) of the Act, I hereby grant a nonexclusive authorization (license) to utilize this thesis, including any and all computer programs incorporated therein or attached thereto and all corresponding documentation (hereinafter collectively referred to as the "Work"), to any and all persons that wish to utilize the Work. Such persons are entitled to use the Work in any way (including for-profit purposes) that does not detract from its value. This authorization is not limited in terms of time, location and quantity. However, all persons that makes use of the above license shall be obliged to grant a license at least in the same scope as defined above with respect to each and every work that is created (wholly or in part) based on the Work, by modifying the Work, by combining the Work with another work, by including the Work in a collection of works or by adapting the Work (including translation), and at the same time make available the source code of such work at least in a way and scope that are comparable to the way and scope in which the source code of the Work is made available.

In Prague on June 27, 2018 .....

Czech Technical University in Prague
Faculty of Information Technology

Citation of this thesis

Jakovcheski, Bogoljub. *Domain-specific Named Entity Recognition.* Master's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2018.

Abstrakt

Technologie Named Entity Recognition (NER) je i přes neustálý vývoj populární jak v akademické, tak v průmyslové sféře, a to i přes to, že coarse grain (hrubé) použití je častější než fine grained (jemné). V této práci používáme sady dat DBpedia NIF. Zpracováváme je a připravujeme nové sady dat pro trénování modelů se Stanford NER. Experimenty jsou prováděny s trénovanými modely, které pokrývají dopad výsledků při použití globálního a specifického doménového modelu. Další experimenty zkoumají dopad počtu článků použÃŋvaných pro trénování modelů. Výsledky experimentů ukazují, že doménově specifické fine grain modely poskytují lepší výsledky než doménově specifické coarse grain modely i globální modely v obou anotacích. Také modely trénované za použití většího množství článků poskytují lepší výsledky než modely trénované s nižšími počty článků.

Klíčová slova Otevřena Data, Named Entity Recognition, Přirodní zpracování jazyků, NLP, DBpedia, NIF, RDF, SPARQL

Abstract

The popular but still under development Named Entity Recognition (NER) technology has seen a significant usage in both academic and industrial sphere

inspite of it's more dominant coarse grain usage compared to it's fine grain usage. In this thesis, we use DBpedia NIF dataset. We process them, and prepare new datasets ready for training models with Stanford NER. Experiments are provided with trained models which cover the impact of results when used with global domain model and domain specific model. In addition, the experiments examine the impact of number of articles used to train models. The results from the experiments show that the domain specific fine grain models provide a better results than domain specific coarse grain models and global models in both annotations. As well, models trained with higher number of articles give better results than models trained with lower number of articles.

Keywords Open Data, Named Entity Recognition, Natural Language Processing, DBpedia, NIF, RDF, SPARQL

Contents

List of Figures

List of Tables

Introduction

Motivation

Named Entity Recognition (NER)[1] is NLP technique for locating and classifying named entities in text into some pre-defined categories such as locations, organizations, person name, sport etc. Today NER is used to different areas from full-text search and filtering to preprocessing tool for other Natural Language Processing (NLP tasks), such as Text Summarization, Machine Translation, Part-of-speech tagging, Entity linking, Text simplification etc. [2].

Most NER applications are trained on a general text and on a specific domain, the problem is that they are optimized for the specific type of data i.e. specific domain. That means that those NER applications can give very good results on texts or domains that are trained, but bad results for texts on a specific domain for which that NER application is not trained.

Most of the NER applications are trained on a small number of types. For example, at the moment of writing this thesis, Stanford NER[1] has a model that have maximum 7 types, DBpedia Spotlight[2] has model with 31 types, spaCy[3] build-in model has 18 types and spaCy Wikipedia scheme model have 4 types.

The main goal of this thesis is to research possibilities of training NER models for a specific domain and as well for a specific types. To achieve this goal it is necessary to create datasets for certain domains. This research is focused on 3 domains, "POLITICS", "SPORT" and "TRANSPORTATION". Every domain is specified with a certain number on types from DBpedia Ontology, then for creating datasets is used DBpedia NIF dataset who provides data for every Wikipedia article.

[1] http://nlp.stanford.edu:8080/ner/
[2] https://www.dbpedia-spotlight.org/demo/
[3] https://spacy.io/usage/linguistic-features

Goals of the thesis

Vast majority of the developed NER systems have been developed as general-purpose systems. While they can perform well on multiple domains (macro level), on specific domains (micro level) their performance quality might be low. The ultimate goal of the thesis is to develop domain-specific NER models. Guidelines:

- Investigate possible datasets for domain-specific training of NER.
- Develop NER training datasets for several selected domains (e.g. sports, politics, music, etc.).
- Train domain-specific NER model using StanfordNER.
- Validate and evaluate the developed domain-specific NER models.

Thesis outline

This thesis is divided into three chapters.

The first chapter describes the frequently used techniques and the basic concept in Named-Entity Recognition and on the related work is covered existing solutions and already provided experiments in domain specific field.

Second chapter defines the process of pre-processing raw big data to data ready to use for creating datasets. As well the process of choosing the domains and their types. How after defining the domains, data are prepared to datasets ready for using in Stanford NER. And as well how those datasets are used in application to train a models.

The final chapter goes through all experiments that we provided with the created datasets and models, to ensure our goals. As well chapter classifies the impact of the number of abstracts used to train and test different models.

Chapter 1

Background and related work

1.1 Background

1.1.1 Information extraction

Information extraction first appears in late 1970s within NLP field[4]. Information extraction (IE) [3] is the task of automatically extracting structured information from unstructured and/or semi-structured machine-readable documents. In most of the cases, this activity concerns processing human language texts by means of natural language processing (NLP). Recent activities in multimedia document processing like automatic annotation and content extraction out of images/audio/video could be seen as information extraction.

Another view of that what Information extraction is that automatically building a relational database from information contained in unstructured text[4].

To understand better what IE is let's give trivial example[5]. Imagine receiving an email message with some date in it. So extracting date information from mail message, and adding to Calendar to create some event is part of IE. Millions of people use this on their daily basis and they are not aware of that how that works and what technology is used for that.

Figure 1.1 gives us a closer look at what Information extraction (IE) is, and how State-of-the-Art algorithms transform unstructured text to structured sequences understandable for machines.

[4]https://www.slideshare.net/rubenizquierdobevia/information-extraction-45392844 slide 4 of 69

[5]https://ontotext.com/knowledgehub/fundamentals/information-extraction/

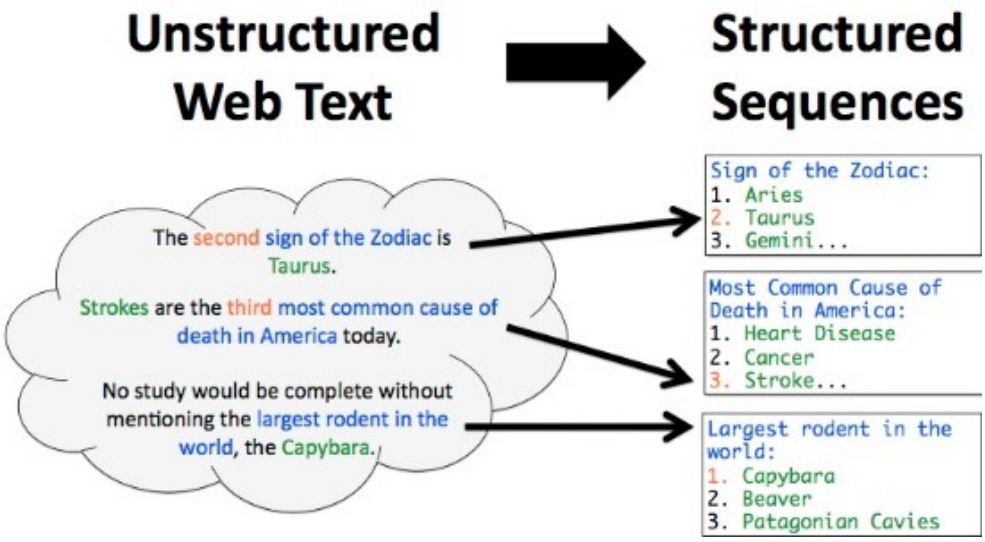

Figure 1.1: Free unstructured texts, parsed and structured with help of IE, downloaded from[6].

1.1.2 Named Entity Recognition

Named Entity Recognition (NER) [5] is the problem of identifying and classifying proper names in text, including locations, such as China; people, such as George Bush; and organizations, such as the United Nations. The named-entity recognition task is, given a sentence, first to segment which words are part of entities, and then to classify each entity by type (person, organization, location, and so on). The challenge of this problem is that many named entities are too rare to appear even in a large training set, and therefore the system must identify them based only on context.

Most research on NER systems has been structured as taking an unannotated block of text, such as this one:

Jim bought 300 shares of Acme Corp. in 2006.

And producing an annotated block of text that highlights the names of entities:

[Jim]Person bought 300 shares of [Acme Corp.]Organization in [2006]Time.

In this example, a person name consisting of one token, a two-token company name and a temporal expression have been detected and classified [1].

Figure 1.2 shows how one NER application can look like. The text in the example is predefined in Stanford NER application and loaded model

[6] `https://www.slideshare.net/rubenizquierdobevia/information-extraction-45392844`

(Classifier) is also trained by Stanford NER[7].

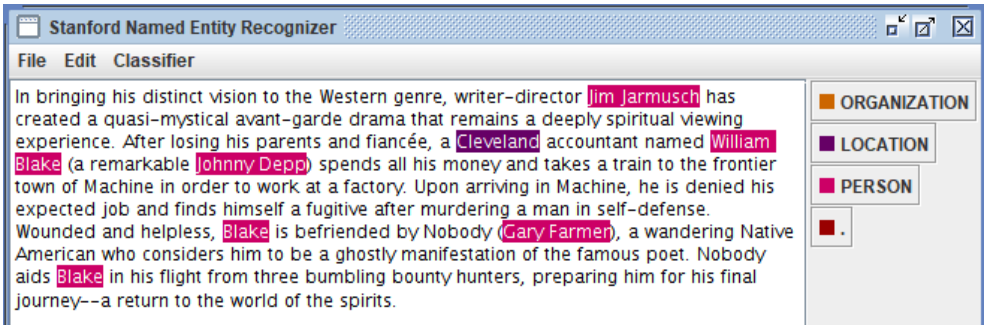

Figure 1.2: Stanford NER GUI with 3 classes model (Location, Person, Organization)

There are several applications or frameworks for NER such as Stanford NER 1.1.2.1, DBpedia Spotlight 1.1.2.2, spaCy 1.1.2.3, Chatbot NER 1.1.2.6, GATE 1.1.2.4, OpenNLP 1.1.2.5 etc. Here we will take a look only on the mentioned ones.

1.1.2.1 Stanford NER

Stanford NER[8] is a Java implementation of a Named Entity Recognizer. Named Entity Recognition (NER) labels sequences of words in a text which are the names of things, such as person and company names, or gene and protein names. It comes with well-engineered feature extractors for Named Entity Recognition, and many options for defining feature extractors. In provided implementation are named entity recognizers for English, particularly for the 3 classes (PERSON, ORGANIZATION, LOCATION).

Stanford NER is implemented as CRFClassifier. The software provides a general implementation of (arbitrary order) linear chain Conditional Random Field (CRF) sequence models. That is, by training your own models on labeled data, you can actually use this code to build sequence models for NER or any other task [6].

1.1.2.2 DBpedia Spotlight

DBpedia Spotlight[9] [7] is a tool for annotating mentions of DBpedia resources in text. This allows linking unstructured information sources to the Linked

[7]https://nlp.stanford.edu/software/CRF-NER.html#Models
[8]https://nlp.stanford.edu/software/CRF-NER.html
[9]https://www.dbpedia-spotlight.org/

Open Data cloud through DBpedia. DBpedia Spotlight performs named entity extraction, including entity detection and name resolution (in other words, disambiguation). It is used for named entity recognition, and other information extraction tasks. DBpedia Spotlight aims to be customizable for many use cases. Instead of focusing on a few entity types, the project strives to support the annotation of all 3.5 million entities and concepts from more than 320 classes in DBpedia. The project started in June 2010 at the Web Based Systems Group at the Free University of Berlin.

1.1.2.3 spaCy

spaCy[10] [8] is an open-source software library for advanced Natural Language Processing, written in the programming languages Python and Cython. It offers the fastest syntactic parser in the world. The library is published under the MIT license and currently offers statistical neural network models for English, German, Spanish, Portuguese, French, Italian, Dutch and multi-language NER, as well as tokenization for various other languages.

1.1.2.4 GATE

General Architecture for Text Engineering or GATE[11] [9] is a Java suite of tools originally developed at the University of Sheffield beginning in 1995 and now used worldwide by a wide community of scientists, companies, teachers and students for many natural language processing tasks, including information extraction in many languages.

GATE includes an information extraction system called ANNIE (A Nearly-New Information Extraction System)[12] which is a set of modules comprising a tokenizer, a gazetteer, a sentence splitter, a part of speech tagger, a named entities transducer and a coreference tagger. ANNIE can be used as-is to provide basic information extraction functionality, or provide a starting point for more specific tasks.

GATE as well has support for NER, for instance StringAnnotation GATE plugin, which is the extended version of ANNIE.

1.1.2.5 OpenNLP

The Apache OpenNLP library[13] is a machine learning based toolkit for the processing of natural language text. It supports the most common NLP tasks, such as tokenization, sentence segmentation, part-of-speech tagging, named entity extraction, chunking, parsing, and coreference resolution. These tasks are usually required to build more advanced text processing services.

[10] https://spacy.io/
[11] https://gate.ac.uk/
[12] http://services.gate.ac.uk/annie/
[13] http://opennlp.apache.org/docs/1.8.4/manual/opennlp.html#intro.description

OpenNLP also included maximum entropy and perceptron based machine learning.

The goal of the OpenNLP project will be to create a mature toolkit for the abovementioned tasks. An additional goal is to provide a large number of pre-built models for a variety of languages, as well as the annotated text resources that those models are derived from.

1.1.2.6 Chatbot NER

Chatbot NER[14] is heuristic based that uses several NLP techniques to extract necessary entities from chat interface. In Chatbot, there are several entities that need to be identified and each entity has to be distinguished based on its type as a different entity has different detection logic.

1.1.3 RDF and NLP interchange format

The Resource Description Framework (RDF)[10] is a family of World Wide Web Consortium (W3C) specifications originally designed as a metadata data model. It is a framework for describing resources on the web; it is designed to be read and understood by computers.

The information in RDF is represented by subject-predicate-object, known as triples. Triples are written in one of RDF notations: RDF/XML, RDFa, N-Triples, Turtle, JSON-LD and as one of the possibility to store those triples is triplestore [11], which we are using in this thesis.

RDF [12] has features that facilitate data merging even if the underlying schemas differ, and it specifically supports the evolution of schemas over time without requiring all the data consumers to be changed.

In this thesis we use datasets that are stored in NIF, which is described below.

Natural Language Processing Interchange Format (NIF)[15] [13] is an RDF-based format. The main idea of NIF is to allow NLP tools to exchange annotations about text in RDF. So the prerequisite is that text should be referencable by URIs, so that they can be used as resources in RDF statements [14]. The structure of the NIF document is round up with nif:Section and nif:Paragraph classes.

The classes to represent linguistic data are defined in the NIF Core Ontology. The NIF Core Ontology also provide properties to describe the relations between substrings, text, documents and their URI schemes [14]. All ontology classes are derived from the main class nif:String which represents strings of Unicode characters.

NIF is built upon the Unicode Normalization Form C, which follows recommendation of the RDF standard for rdf:Literal. Each URI scheme is a

[14]https://haptik.ai/tech/open-sourcing-chatbot-ner/
[15]http://aksw.org/Projects/NIF.html

subclass of nif:String. Users of NIF can also create their own URI schemes by subclassing nif:String and providing documentation on the Web in the rdfs:comment field. This puts restriction over the URI syntax, so for example instances of type nif:RFC5147String have to adhere to the NIF URI scheme based on RFC 5147.

As well another important subclass of nif:String is the nif:Context OWL class. This class is assigned to the whole string of the text. The purpose of an individual of this class is special, because the string of this individual is used to calculate the indices for all substrings. Therefore, all substrings have to have a relation nif:referenceContext pointing to an instance of nif:Context. Listing 1.1 provides an example of NIF structure.

```
prefix rdf:<http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
prefix xsd:<http://www.w3.org/2001/XMLSchema#> .
prefix itsrdf:<http://www.w3.org/2005/11/its/rdf#> .
prefix nif:<> .
prefix ex:<http://nif.dbpedia.org/wiki/en/> .

ex:United_States?dbpv=2016-10&nif=context a nif:Context;
nif:beginIndex "0"^^xsd:nonNegativeInteger;
nif:endIndex "104211"^^xsd:nonNegativeInteger;
nif:firstSection ex:United_States?dbpv=2016-10&char=0,4241;
nif:lastSection ex:United_States?dbpv=2016-10&char=103211,
104211;
nif:hasSection ex:World_War_II?dbpv=2016-10&char=0,5001;
nif:sourceUrl ex:United_States?oldid=745182619;
nif:predLang <http://lexvo.org/id/iso639-3/eng>;
nif:isString "...The first inhabitants of North America
migrated from Siberia by way of the Bering land bridge..." .

ex:United_States?dbpv=2016-10&char=7745,9418 a nif:Section;
nif:beginIndex "7745"^^xsd:nonNegativeInteger;
if:endIndex"9418"^^xsd:nonNegativeInteger;
nif:hasParagraph ex:United_States?dbpv=2016-10&char=7860,8740;
nif:lastParagraph ex:United_States?dbpv=2016-10&char=8741,9418;
nif:nextSection ex:United_States?dbpv=2016-10&char=9420,12898;
nif:referenceContext ex:United_States?dbpv=2016-10&nif=context;
nif:superString ex:United_States?dbpv=2016-10&char=7548,7743 .

ex:United_States?dbpv=2016-10&nif=paragraph&char=7860,8740
anif:Paragraph;
nif:beginIndex"7860"^^xsd:nonNegativeInteger;
nif:endIndex"8740"^^xsd:nonNegativeInteger;
nif:nextParagraphex:United_States?dbpv=2016-10&char=8741,9418;
nif:referenceContextex:United_States?dbpv=2016-10&nif=context;
nif:superString ex:United_States?dbpv=2016-10&char=7745,9418.

ex:United_States?dbpv=2016-10&char=7913,7920 a nif:Word;
```

```
nif:anchorOf"Siberia";
nif:beginIndex"7913"^^xsd:nonNegativeInteger;
nif:endIndex"7920"^^xsd:nonNegativeInteger;
nif:referenceContext ex:United_States?dbpv=2016-10&nif=context;
nif:superString ex:United_States?dbpv=2016-10&char=7860,8740;
itsrdf:taIdentRef<http://dbpedia.org/resource/Siberia> .
```

Listing 1.1: Example of NIF taken from [16]

1.1.4 DBpedia

DBpedia [15] is a crowd-sourced community effort to extract structured content from the information created in various Wikimedia projects. This structured information resembles an open knowledge graph (OKG) which is available for everyone on the Web. A knowledge graph is a special kind of database which stores knowledge in a machine-readable form and provides a means for information to be collected, organised, shared, searched and utilised. Google uses a similar approach to create those knowledge cards during search.

DBpedia data is served as Linked Data, which is revolutionizing the way applications interact with the Web. One can navigate this Web of facts with standard Web browsers, automated crawlers or pose complex queries with SQL-like query languages (e.g. SPARQL).

At the time of writing this thesis the last version of DBpedia is 2016/10.

1.1.4.1 DBpedia NIF

DBpedia [16] currently primarily focus on representing factual knowledge as contained in Wikipedia infoboxes. A vast amount of information, however, is contained in the unstructured Wikipedia article texts. DBpedia NIF also considers to broad and deep the amount of structured data.

With the representation of wiki pages in the NLP Interchange Format (NIF) are provided all information directly extractable from the HTML source code divided into three datasets:

- nif-context: the full text of a page as context (including begin and end index)
- nif-page-structure: the structure of the page in sections and paragraphs (titles, subsections etc.)
- nif-text-links: all in-text links to other DBpedia resources as well as external references

[16] https://2018.eswc-conferences.org/wp-content/uploads/2018/02/ESWC2018_paper_136.pdf

These datasets will serve as the groundwork for further NLP fact extraction tasks to enrich the gathered knowledge of DBpedia.

For the purposes of this thesis we will use English version of DBpedia NIF dataset version 2016-04 (dbpv=2016-04).

1.1.4.2 DBpedia ontology

The DBpedia Ontology is a shallow, cross-domain ontology, which has been manually created based on the most commonly used infoboxes within Wikipedia. The ontology currently covers 685 classes which form a subsumption hierarchy and are described by 2,795 different properties.

Since the DBpedia 2016/10 release, the ontology is a directed-acyclic graph, not a tree. Classes may have multiple superclasses, which is important for the mappings to schema.org. [17].

DBpedia ontology classes can be found here[17].

The DBpedia Ontology version 2016-10 currently contains about 4,233,000 instances only in English. Figure 1.3 shows the number of instances for several classes within the ontology[18].

Class	Instances
Resource (overall)	4,233,000
Place	735,000
Person	1,450,000
Work	411,000
Species	251,000
Organisation	241,000

Figure 1.3: DBpedia Ontology - Instances per class

1.1.5 Apache Jena

Apache Jena[19] [18] is an open source Semantic Web framework for Java. It provides an API to extract data from and write to RDF graphs. The graphs

[17] http://mappings.dbpedia.org/server/ontology/classes/
[18] http://wiki.dbpedia.org/services-resources/ontology
[19] https://jena.apache.org/index.html

are represented as an abstract "model". A model can be sourced with data from files, databases, URLs or a combination of these. A Model can also be queried through SPARQL 1.1.

1.1.6 SPARQL

SPARQL [11] is an RDF query language, that is, a semantic query language for databases, able to retrieve and manipulate data stored in Resource Description Framework (RDF) format. SPARQL works for any data source that can be mapped to RDF.

SPARQL allows users to write queries against key-value data or, more specifically, data that can be mapped to RDF. The entire database is thus a set of subject-predicate-object triples.

The SPARQL standard [20] is designed and endorsed by the W3C and helps users and developers focus on what they would like to know instead of how a database is organized.

In Listing 1.2 is shown an example of SPARQL query where we are selecting 10 abstracts (articles) from DBpedia NIF who has ontology type PoliticalParty and their PageRank value. The results are sorted in descending order, according to their PageRank value.

```
PREFIX rdf:<http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX dbo:<http://dbpedia.org/ontology/>
PREFIX vrank:<http://purl.org/voc/vrank#>

SELECT DISTINCT ?s ?v
FROM <http://dbpedia.org>
FROM <http://people.aifb.kit.edu/ath/#DBpedia_PageRank>
WHERE{
        ?s rdf:type dbo:PoliticalParty .
        ?s vrank:hasRank/vrank:rankValue ?v.
}
ORDER BY DESC(?v) LIMIT 10
```

Listing 1.2: SPARQL example

1.2 Related work

Traditionally Named Entity Recognition (NER)[19] systems have been built using available annotated datasets (like CoNLL, MUC) and demonstrate excellent performance. However, these models fail to generalize onto other domains like Sports and Finance where conventions and language use can differ

[20] https://ontotext.com/knowledgehub/fundamentals/what-is-sparql/

significantly. Furthermore, several domains do not have large amounts of annotated labeled data for training robust Named Entity Recognition models. With specifying the domain we can create a bigger model with more annotated words and reading the whole text will be same or even faster that reading text with a global domain.

In [20] authors used the WordNet English database for retrieving entities. As domain they chose Lord of the Rings book and "LOCATION" and "PERSON" as types. After the small experiment they figure out that the "PERSON" type is quite a large, so after some fine-tuning they changed to "ANIMATE" type, which in their main experiment gives better results.

Authors in [21] propose methods to effectively adapt models learned on one domain onto other domains using distributed word representation from Online Media. As well they demonstrate how to effectively use such domain specific knowledge to learn NER models. They chose "FINANCE" and "SPORT" types because as they say domains from CoNLL or MUC performs poor results. Also are compared the global model and a domain specific models, so from the observation the training data in domain specific models get increased and perform better results.

As well [22] authors for domain chose tweets and use T-NER system, and for comparing the results, they used Stanford NER system. They created a dataset with 2400 annotated tweets with 10 popular tweets domain. Based on their experiments the T-NER system performs a way more better results than the Stanford NER system, where they have 3 types, "PERSON", "LOCATION" and "ORGANIZATION".

Chapter 2

Domain specific named entity recognition

In this chapter, as we can see from Figure 2.1 we will go through the whole process of transforming raw DBpedia datasets to datasets that are ready for training a models with Stanford NER and the process of training models with Stanford NER. Section 2.1 explains the process of filtering the data with relevant information from DBpedia NIF datasets and preparing them for processing. In Section 2.2 we explain how we choose "POLITICS", "SPORT" and "TRANSPORTATION" domains. Section 2.3 shows all ontology types that we retrieve for every domain and grouping them to more specific ontology type. In Section 2.4 is explained the process of preparing datasets for training in Stanford NER. And finally in Section 2.5 is shown the process of training models from prepared datasets with Stanford NER.

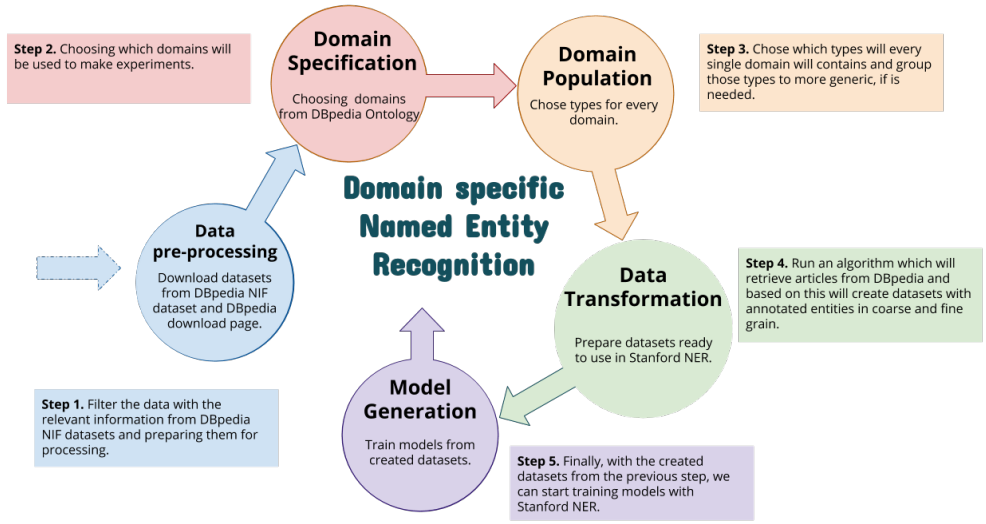

Figure 2.1: Chapter 2 flow

2.1 Data pre-processing

To be able to create domain specific datasets we need training data which covers multiple domains, where used multiple types. We choose data from DBpedia NIF Datasets (for more information about DBpedia NIF see Section 1.1.4.1) for the English language in .ttl format. This dataset is provided with 3 partitions that we used only 2 of them. Those partitions are:

- **nif-context**: This partition contains full text of a Wikipedia page as context (including begin and end index)
- **nif-text-links**: This partition contains all in-text links to other DBpedia resources also external references

Because DBpedia NIF dataset does not contains entity types dataset, we should use instance-types_en datasete[21], who is also in .ttl format. This dataset contains all types of nif-text-links that occurrence at nif-abstract-context file.

So how all this dataset are connected between themselves? Let say that we have abstract for Alexander the Great. In nif-text-links file we have all words from the abstract that have annotation, but we still don't know their type. So here comes instance-types_en dataset where based on link from nif-text-link (eg.http://dbpedia.org/resource/Philip_II_of_Macedon) we can find the type of annotated word (word Philip II has ontology type Monarch), but of course,

[21] `http://wiki.dbpedia.org/downloads-2016-04`

there might be a case that some words cannot be found on instance types file and automatically have no type, or in our case ontology type O (O stands for OTHER).

Let us explain in detail how we process and clean data from the datasets. First, we define small test dataset to check how fast we can process data. Processing that dataset on downloaded files without any cleaning of data takes too long. In order to improve processing speed of the dataset we consider converting data RDF format to binary format (.ttl to .hdt) with RDF/HDT tool[22] will be faster. HDT (Header, Dictionary, Triples)[23] is a compact data structure and binary serialization format for RDF that keeps big datasets compressed to save space while maintaining search and browse operations without prior decompression. So we converted the datasets and reran the data processing face again. There were some processing time improvements, but not satisfying for our purposes. Because we don't need all information that datasets contains, our next solution was to clean datasets from those unused data for our aims. The final result after cleaning was a smaller datasets, for instance, nif-abstract-context file from 7.78GB now has 2.99GB, another big improvement was nif-text-links file who is reduced to 10.5GB from 44.6GB and at the end we also clean instance-types file, but here we don't record any major memory improvements. Again we rerun the algorithm, of course, there were improvements, but as well as previous the processing time was not acceptable. To give an illustration, the time needed to find all types from one abstract in a worst case, to read nif-text-links and instance type files until the end was around 3.5 minutes. Therefore we converted our cleaned datasets from RDF format(.ttl) to binary format(.hdt). And how in previous processing face there were again time improvements, but those improvements didn't meet our criteria. So we decided not to use binary format, but we will create dataset tree only for nif-text-links and instance-types_en datasets.

The reason why we chose to create a tree is, because, who we see previously, reading a big dataset takes a lot of time and memory as well. So diving that dataset to smaller pieces will reduce the memory usage and reading time, which as well will reduce the processing time of algorithm. We create a dataset tree for nif-text-links and instance-types datasets. For nif-text-links dataset we created a tree where we have folders from "a-z", also special characters folders and other folder (this folder contains data that have a lower occurrence, let say & character or letters that are not part of the English alphabet) and folders from "a-z" has subfolders also from "a-z".

To give a closer look how we create that tree, let say that we have an abstract for Volkswagen Golf MK3, so the link for that abstract would be http://dbpedia.org/resource/Volkswagen_Golf_Mk3 and this link will be stored to "v" folder and "o" subfolder, because the title of the abstract is Volkswagen Golf MK3, where we need only first 2 letters from the first word, in this case

[22] `http://www.rdfhdt.org/`

word Volkswagen. As we say earlier, this will create smaller dataset, where we need less time to read it.

For instance-types_en dataset we modified the process of creating a data tree datasets. Here because of lower range data we have created only datasets from "a-z", of course, special characters dataset and dataset with a special characters in the beginning of their names that have lower occurrence, for example & or ´ character etc.

Finally, we rerun the algorithm, and the time to process one abstract, at worst case, takes no longer than 1 minute. Now we were ready to take next steps to create domains (see Section 2.2), retrieve types (see Section 2.3) and prepare data for Stanford NER (see Section 2.4).

2.2 Domain specification

As we said earlier most of the NER application are trained on same domains, like "PERSON", "ORGANIZATION" and "LOCATION". These 3 domains are widely spread all over the applications and perform nice results on text from these domains. So what we need is something that is not already trained or there is a small usage of that domain. After some research, we find out that "TRANSPORTATION" domain is not a popular domain for NER applications, respectively in time of writing the thesis we don't find any usage of this specific domain. So there is the possibility to create this specific domain. Types that we retrieve for this domain and groping them to more specific types are more deeply explained in Types retrieval (see Section 2.3). We have our first domain, but at least 2 more domains are needed to be able to make some experiments and conclusion.

The next domain that we chose was "POLITICS". The reason why we chose "POLITICS" domain is because it is widely covered in DBpedia, which gives an opportunity for quality working and testing with that domain. The types that contains this domain are explained in Section 2.3. The second domain is chosen, so we need at least one more domain to keep up with other NER applications.

It was not an easy task to select a domain having in mind the criteria we set. After a research, also referring to ontology types from previous two domains and some NER applications (see Section 1.1.2) we find an opportunity to create the last "SPORT" domain. Now we should check on DBpedia ontology classes page (see Section 1.1.4.2) how many ontology types we have for this domain. At the time of writing this thesis there were around 170 ontology types, which is very good number for creating a domain (for more see Section 2.3).

After we complete choosing of domains, the next step was to choose the right ontology types for every specific domain and if it is needed or make

sense group those types to more specific type. This is explained in detail at Section 2.3.

2.3 Domain population

After the problem of running the algorithm to find all types from the abstract and choose domains, next issue was which types we want to be part of our domains and also which types we want to retrieve from DBpedia. Worth mentioning that we will use the same ontology types for retrieving the abstracts links from DBpedia and creating a domain models. For example the type "Politician" will be used to retrieve links from DBpedia that has that type, and also "Politician" type will be use to annotated words, for instance Barack Obama will have type of "Politician" (we will give more details on section 2.4).

In DBpedia ontology classes page[23] we can see all types that DBpedia ontology has. Those ontology types are the same in instance types file also. Now we are facing with the fact that if we choose very small group of ontology types, at the experiment point we will have minor range of annotated words and experiments won't be relevant. On the other hand, if we go too deep to ontology types, we will have a lot of annotated words, which might be positive, but training the model will take a lot of time and memory. There is a possibility that we will reach memory exception, or because of big group of types training will never end.

After some testing with the number of retrieved types we finally found the best selection of types, in total we choose 283 ontology types for all domains.

Now let us explain more deeply every single domain and which types has that domain. We have 3 domains (see Section 2.2 for that how we choose those domains) "POLITICS", "SPORT" and "TRANSPORTATION".

In "POLITICS" domain we retrieve in total 26 types, found at Appendix A.2, which we sort in 11 more specific types like Ambassador, Chancellor, Congressman, Deputy, Governor, Lieutenant, Mayor, MemberOfParliament, Minister, President, PrimeMinister, Senator, VicePresident and VicePrimeMinister are joined together in one specific domain Politician, other types we leaved as it is, because if we group them the types wouldn't give any sense.

We do the same for "SPORT" domain where we retrieve in total 171 types, found in Appendix A.3, so those types, same as "POLITICS" domain, are more specified in 8 types, like SportClub, SportsLeague, SportsTeam, Athlete, Coach, OrganizationMember, SportsManager and SportsEvent. Grouping of types is also shown in appendix A.3. This domain is a nice example of that even we retrieve quite a big number of types, we can reduce that number with more specific types which don't lose the meaning further. For instance "David de Gea" has a type of SoccerPlayer, but after processing will have type of Athlete, which gives sense, because any type of sport player is an athlete.

[23] `http://mappings.dbpedia.org/server/ontology/classes/`

At the end we repeat the process for "TRANSPORTATION" domain, where we retrieve in total 86 types. Retrieved types can be found in Appendix A.4. Those types are after minimized in 14 more specific types like Aircraft, Automobile, On-SiteTransportation, Locomotive, MilitaryVehicle, Motorcycle, Rocket, Ship, SpaceShuttle, SpaceStation, Spacecraft, Train, PublicTransitSystem and Infrastructure. The logic of that who we create more specific ontology types is same as in "POLITICS" or "SPORT" domain.

The reason why we group ontology types to more generic ones is that, that when the dataset has a smaller number of types, training a model with Stanford NER is more faster and requires less memory for training. Another reason is faster providing a NER, because is needed to read less types and also the overall results after testing with same data perform better than when ontology types where not grouped.

2.4 Data transformation

We define domains as well their types that we will retrieve and process, now we should put everything together and prepare data for Stanford NER application. In Data pre-processing (see Section 2.1) we explain how we handled the data downloaded from DBpedia NIF dataset and we briefly touch how those data will be prepared for training in Stanford NER application.

The final thing that is missing is how we will choose which abstracts will be part of our models. Because our goal is to create models with different number of abstracts we need some strict order of retrieved links from DBpedia dataset. In order to chose the data, we chose those articles with higher PageRank values. PageRank [24] is an algorithm used by Google Search to rank websites in their search engine results. So with a prepared SPARQL queries (SPARQL queries for every domain can be found in A.6, A.7 and A.8 appendix) and with help of Apache Jena framework (see Section 1.1.5) we implemented retrieving links, on Java, on DBpedia endpoint[24]. After retrieving those data, based on their PageRank we check does retrieved link from DBpedia is part on our abstract file (nif-context dataset). If link is found in nif-context dataset it's written to two files, one file is where are written all abstracts from every domain and another file is file for that specific domain. Those files are creating in RDF format, with n-triples, that means that there is subject, in our case that is the link of abstract, then predicate who has isString annotation which tells that next triple contains the abstract text and finally object where abstract text is placed. Next thing that we need to do is to find all annotated words from abstract and their types. The algorithm of finding types is explained in Section 2.1. What is not mention there is that after finding the types, the abstract is written to file, where on first position is word and on the second position is the type of that word, if there is any,

[24] `http://www.dbpedia.com/sparql`

if not the type is O. Final step is to prepare data to be able to train models in Stanford NER with the types that we define in Section 2.3. Because files contains all types that were found on the abstracts we need to clean and group them, as well to create datasets with coarse and fine grained entities with fine and coarse types. The algorithm is very simple, it reads the files which already have all types and if type is part of our retrieved types then either type is leaved as it is, or is grouped to more specific type, for instance if word has type Ambassador, then after filtering that word will have Politician type. The same is for coarse grained annotation, but here proper types after filtering are "POLITICS", "SPORT" or "TRANSPORTATION" type. The whole process is also illustrated at Algorithm 1.

Interesting fact is that, that when we retrieve links from DBpedia with a specific ontology types types, some links there have types that are not even part of our domain. Here are some interesting links that we catch:

- http://dbpedia.org/page/Orbital_period
- http://dbpedia.org/page/Pregnancy
- http://dbpedia.org/page/Melody
- http://dbpedia.org/page/ITunes
- http://dbpedia.org/page/Tachycardia
- http://dbpedia.org/page/Shortwave_radio
- http://dbpedia.org/resource/UTC-05:00

```
Retrieve links from DBpedia NIF Dataset based on their PageRank;
if Retrieved link is found at nif-abstract dataset then
    write value from nif-abstract dataset to file
else
    go to next retrieved link and repeat steps
end
Read new file with values from nif-context and get abstract links;
Check does that link is consists in nif-text-links dataset;
if link consists in nif-text-links then
    Get all values (links) from nif-text-links dataset;
    Search for ontology types in instance-types dataset;
    if Link from nif-text-links exists in instance-types then
        Parse value and return ontology type;
    else
    end
    Write abstract text to domain specific file with founded type of
     the word, as only word and the type at a line;
else
    Write abstract text to domain specific file with word and O type
     at a line;
end
Read created domain specific files and clean unnecessary types;
if Type equals some of retrieved types then
    Leave type as it is or group type and write to two domain specific
     files in coarse and fine grained;
else
    Rewrite the type to "O" and write to two domain specific files in
     coarse and fine grained;
end
Write to two domain specific files in coarse and fine grained;
```

Algorithm 1: Algorithm for preparing datasets ready for training in Stanford NER

2.5 Model generation

With the created files from Section 2.4 now we can start training models. At Stanford NER CRF FAQ webpage[25] provides explanation of that how to train own model with Stanford NER. We follow those steps and used the same NER properties file with a small correction where we had to add 2 more flags to be able to train big models. Those two flags are saveFeatureIndexToDisk=true, which is used on every properties file and for creating a models in fine grained we use useObservedSequencesOnly=true. Flag saveFeatureIndexToDisk stands for saving the feature name's to disk that aren't actually needed while the core model estimation (optimization) code is run. Another flag that we use is useObservedSequencesOnly flag. It is used for labeling only adjacent words with label sequences that were seen next to each other in the training data. For some kinds of data this actually gives better accuracy, for other kinds it is worse. After testing on a small model with only 40 abstracts and model with 300 abstracts we find out that for creating a fine grained model with 40 and more abstract this flag gives us better results, while on coarse grained models this flag gives worst results, the exception are models with 500 abstracts where we should use this flag to reduce memory usage. The whole properties file with all used flags can be found in Appendix A.5.

After creating a properties files, training models is very easy with only one command, where unlike command from Stanford we add Xmx Java option, because standard command use only 4GB of RAM, which for our purposes is not enough for training big models.

Command for training model ran from the stanford-ner folder:

```
java -Xmx11g -cp stanford-ner.jar
edu.stanford.nlp.ie.crf.CRFClassifier -prop
locationAndnameOfPropFile.prop
```

2.5.1 Training datasets

For the aim of our experiments we have trained 57 models. As mentioned earlier for training we have used Stanford NER application explained in Section 1.1.2.1. We have two types of models, coarse-grained and fine-grained, also those model types are divided in to "POLITICS", "SPORT" or "TRANSPORTATION" specific domains and a global domain who contains all abstracts from every domain. To give an illustration, for dataset with 100 retrieved abstract we will have 4 coarse-grained models (global domain and 3 specific domains), and similarly for a fine-grained models, so in total we have 8 trained models for every dataset. We created 7 different groups of datasets with 10 abstracts, 20 abstracts, 40 abstracts, 100 abstracts, 300 abstracts, 400 abstracts and 500 abstracts. Each of this datasets has 8 trained models and

[25] https://nlp.stanford.edu/software/crf-faq.html

we have one dataset that have also 500 abstracts, but those abstracts are not the same like the previous dataset. This dataset contains abstracts that have lower PageRank value and has only one trained model with abstracts from every domain in fine grained.

Chapter 3

Experiments

There are parameters of the computer used for tests shown in Table 3.1.

Table 3.1: Testing computer parameters

Part	Description
CPU	2.00 GHz Intel(R) Core(TM) i5-4310U
MEM	16 GB DDR3L
OS	x86_64 Windows 10 Pro
DISK	240GB SSD Kingston

We have provide various types of experiments. In next sections we will discuss more about every provided experiment.

3.1 Goals of the experiments

We set a few goals of the experiments. Those goals are:

- To investigate the impact of coarse grain global model results against the fine grain global model results.
- To evaluate the performance of domain specific models.
- To evaluate the performance of global models.
- To investigate the impact of global models tested with domain specific dataset.

3.2 Evaluation metrics

The success of NER systems is exposed to F_1 score (F-score or F-measure). F_1 [25] score is a measure of a test's accuracy. It considers both the precision

P and the recall **R** of the test to compute the score: P is the number of correct positive results divided by the number of all positive results returned by the classifier, and R is the number of correct positive results divided by the number of all relevant samples (all samples that should have been identified as positive). The F_1 score is the harmonic average of the precision and recall, where an F_1 score reaches its best value at 1 (perfect precision and recall) and worst at 0. Written in formula, the $F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$.

3.3 List of experiments

With our trained models we made a few experiments. First one is the model that has 300 abstract on every domain(900 abstract in total). This is our main model and other experiments that we will provide, models that has lower or higher number of abstracts or experiments where model has more abstracts that a test file or vice-verse, all those results will be compared with the results obtained from main experiment. This experiment can be found an Section 3.3.1.

With the experiments we wanted to answer some important questions:

- What is the impact on results when models are trained with less data than in the main experiment?

  Section 3.3.2 has answer to this question.

- What is the impact on results when models are trained with more data than in the main experiment?

  Section 3.3.3 gives a closer look to this questions.

- What is the impact on results when models are train in fine or coarse grain?

  In both sections (Section 3.3.2 and Section 3.3.3 and as well in Section 3.3.1) we provide those types of experiments.

- What is the impact on results when trained model is tested with more than one dataset?

  Section 3.3.4 has the answer of this question.

- How the models from group of 500 abstracts per domain will perform when are tested with news articles?

  Section 3.3.5 gives a closer look to this question.

Figure 3.1 shows the time that algorithm explained in Section 2.4 need to process data and prepare datasets ready for training with Stanford NER. As we can see time grows approximately linearly.

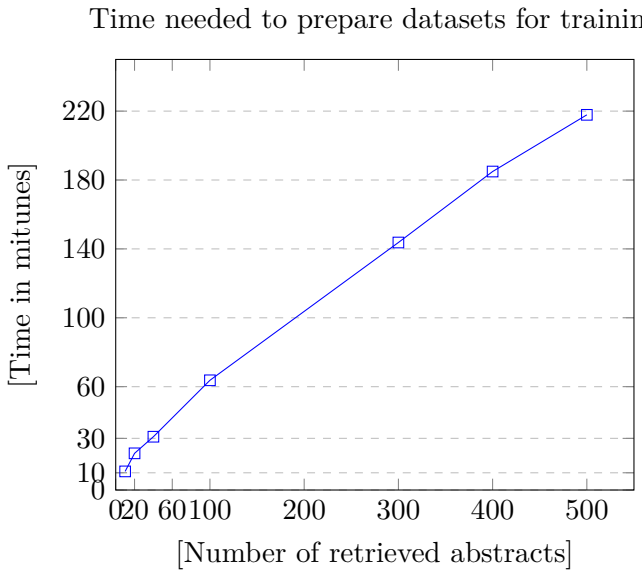

Figure 3.1:

3.3.1 Main experiment

This is our main experiment where other experiments will be compared with this one. This model is trained with top 300 Wikipedia abstracts for every domain. Algorithm for preparing the data for training model explained in Section 2.4 takes 2.40 hours. The model is trained in coarse grained and takes 873.7 seconds, from which 844.63 seconds spent in optimization.

3.3.1.1 Global domain models

First experiment that we do with this model is that we run it with the same text that model is trained in coarse grained. Results are promising, we are above 95% as shown in Table 3.2, which is great number for such quite big model. With such results, someone will say that those are nice results and other experiments will only have the worst results. But let see how model behaves when we test with abstracts for every specific domain.

Entity	**Precision**	**Recall**	**F1 score**
POLITICS	0,9872	0,9462	0,9662
SPORT	0,9846	0,9629	0,9736
TRANSPORTATION	0,9940	0,9823	0,9881
Totals	**0,9875**	**0,9625**	**0,9748**

Table 3.2: Results of base experiment run to be used as reference for subsequential experiments

Table 3.3 shows the output of model when is tested with abstracts from a "POLITICS" domain. As we said in Section 2.4 this type of abstract has the biggest word annotation. Result is not even close with the result from previous experiment. Also, trained model annotated words with a "TRANSPORTATION" domain, where the test file don't have any word with that annotation.

Entity	**Precision**	**Recall**	**F1 score**
POLITICS	0,9839	0,4025	0,5713
TRANSPORTATION	0,0000	1,0000	0,0000
Totals	**0,9792**	**0,4025**	**0,5705**

Table 3.3: Results of base model in coarse grained run with "POLITICS" abstracts

Table 3.4 gives us results from abstracts from "SPORT" domain. Here we have the same results like in first experiment, but because trained model annotated some words with a "POLITICS" or "TRANSPORTATION", even those that our test file contains only abstracts from "SPORT" domains and words has only "SPORT" type, the overall result is only a little bit lower that the first experiment.

Entity	**Precision**	**Recall**	**F1 score**
POLITICS	0,0000	1,0000	0,0000
SPORT	0,9846	0,9628	0,9736
TRANSPORTATION	0,0000	1,0000	0,0000
Totals	**0,9819**	**0,9628**	**0,9722**

Table 3.4: Results of base model in coarse grained run with "SPORT" abstracts

Table 3.5 provide outcome with testing with abstracts only from "TRANSPORTATION" domain. As in the previous experiment, the result now is almost the same like in first experiment, but even thought that trained model, as in previous 2 experiments, annotated words with a "SPORT" type, the overall results is better that the experiment where test file contains all abstracts from every domain.

Entity	**Precision**	**Recall**	**F1 score**
SPORT	0,0000	1,0000	0,0000
TRANSPORTATION	0,9940	0,9822	0,9880
Totals	**0,9861**	**0,9822**	**0,9841**

Table 3.5: Results of base model in coarse grained run with "TRANSPORTATION" abstracts

In conclusion with this kind of experiments we can say that it is not a good idea to train a model with all chosen domains and then use texts from specific domain to perform NER.

After we finish the experiments with model that is trained with all abstracts from every domain in coarse grained, we wanted to see the impact of model that is trained with same abstracts, but now annotated in fine grained. To train this model we needed 3250.9 seconds from which 3207.45 seconds for optimization. Table 3.6 shows the results of provided experiment where we can see that we have a little bit more better total result than experiment in Table 3.2.

Entity	**Precision**	**Recall**	**F1 score**
Aircraft	1,0000	1,0000	1,0000
Athlete	1,0000	0,9802	0,9900
Automobile	1,0000	1,0000	1,0000
Coach	1,0000	1,0000	1,0000
Infrastructure	1,0000	0,9820	0,9909
PoliticalParty	0,9860	0,9628	0,9743
Politician	1,0000	0,9353	0,9665
PublicTransitSystem	0,9919	0,9839	0,9879
Ship	1,0000	1,0000	1,0000
SpaceShuttle	1,0000	1,0000	1,0000
SpaceStation	1,0000	1,0000	1,0000
SportsClub	0,9796	0,9683	0,9739
SportsEvent	1,0000	0,9242	0,9606
SportsLeague	0,9647	0,9805	0,9725
SportsManager	1,0000	0,9423	0,9703
SportsTeam	1,0000	0,9805	0,9902
Train	1,0000	1,0000	1,0000
Totals	**0,9880**	**0,9712**	**0,9795**

Table 3.6: Results of base experiment in fine grained run to be used as reference for subsequential experiments

Then we tested our model with abstracts from ”POLITICS” domain. How we can see from Table 3.7 there is some improvements on overall result unlike the experiment in coarse grained, but no satisfying at all. As well table shows that some words again are annotated with types from ”SPORT” and ”TRANSPORTATION” domain.

Entity	Precision	Recall	F1 score
Election	0,0000	0,0000	0,0000
PoliticalParty	0,9860	0,9628	0,9743
Politician	1,0000	0,1849	0,3120
PublicTransitSystem	0,0000	1,0000	0,0000
Ship	0,0000	1,0000	0,0000
SportsLeague	0,0000	1,0000	0,0000
Totals	**0,9825**	**0,4072**	**0,5758**

Table 3.7: Results of base model in fine grained run with "POLITICS" abstracts

After that we rerun the experiment, but now with abstracts from "SPORT" domain. In Table 3.8 we can see minor growth of the results unlike experiment in Table 3.4, but these improvements are so small that are almost unimportant. Also our model annotated some words with types from "POLITICS" and "TRANSPORTATION" domain which the test file don't have those types at all.

Entity	Precision	Recall	F1 score
Athlete	1,0000	0,9802	0,9900
Coach	1,0000	1,0000	1,0000
Politician	0,0000	1,0000	0,0000
SportsClub	0,9794	0,9680	0,9737
SportsEvent	1,0000	0,9242	0,9606
SportsLeague	0,9678	0,9805	0,9741
SportsManager	1,0000	0,9423	0,9703
SportsTeam	1,0000	0,9804	0,9901
Train	0,0000	1,0000	0,0000
Totals	**0,9821**	**0,9716**	**0,9768**

Table 3.8: Results of base model in fine grained run with "SPORT" abstracts

Finally the last experiment with this model are the abstracts from "TRANSPORTATION" domain. Table 3.9 shows the output of the provided experiment, where like in previous 2 experiments we can notice a very little improvements on results, from experiment in Table 3.5, who again can be unimportant. As in previous experiments similarly here model annotated some words with types from other 2 domains, which test file does not even contain.

Entity	Precision	Recall	F1 score
Aircraft	1,0000	1,0000	1,0000
Automobile	1,0000	1,0000	1,0000
Infrastructure	1,0000	0,9820	0,9909
Politician	0,0000	1,0000	0,0000
PublicTransitSystem	0,9918	0,9837	0,9878
Ship	1,0000	1,0000	1,0000
SpaceShuttle	1,0000	1,0000	1,0000
SpaceStation	1,0000	1,0000	1,0000
SportsClub	0,0000	1,0000	0,0000
SportsTeam	0,0000	1,0000	0,0000
Train	1,0000	1,0000	1,0000
Totals	**0,9862**	**0,9881**	**0,9871**

Table 3.9: Results of base model in fine grained run with "TRANSPORTATION" abstracts

Provided experiments with the model which is trained with all abstracts from every domain annotated in fine grained, overall provide a very little improvement on results on every experiment. With that observation trained model annotated in fine grained is better to use instead of the model that is annotated in coarse grained. Another benefit of this type of model is that we can see which types are annotated and their results. But, because those improvements are small and is needed almost four times more time to train a fine grained model, maybe the better solution will be models trained in coarse grained. Everything depends on our goals: faster models or more precision?

3.3.1.2 Evaluation of domain specific models

After completing experiments with global domains in coarse and fine grained, now we will make experiments with models for specific domains.

To train "POLITICS" domain specific model we need 66.7 seconds in total from which 59.53 seconds spend on optimization. In Table 3.10 the experiment is provided with model trained only with abstracts from "POLITICS" domain and run with the same texts that model in trained, in coarse grained. The result here is better than experiment in Table 3.3, but worse that experiment provided with global domain in Table 3.2. This can be cause by the fact that model has biggest number of annotated words.

Entity	Precision	Recall	F1 score
POLITICS	0,8039	0,6779	0,7355
Totals	**0,8039**	**0,6779**	**0,7355**

Table 3.10: Results of "POLITICS" base model in coarse grained run with "POLITICS" abstracts

We repeated the previous experiment, but now everything in fine grained. Time for training this kind of model in total was 163.5 seconds, from which 155.94 seconds spend on optimization. Table 3.11 shows that this kind of model provides better result that coarse grained model and the experiment provided in Table 3.7, but again worst than model trained with all abstracts (see Table 3.6).

Entity	**Precision**	**Recall**	**F1 score**
Election	0,8240	0,6398	0,7203
PoliticalParty	0,8100	0,7006	0,7513
Politician	0,8599	0,7234	0,7858
Totals	**0,8354**	**0,6980**	**0,7606**

Table 3.11: Results of "POLITICS" base model in fine grained run with "POLITICS" abstracts

In conclusion with provided 2 experiments and from this point of view, for this domain we can say that training a specific model will give better results and will perform faster that global domain tested with text from specific domain. On the other hand the global domain tested with a texts that is trained, how we can see from Table 3.6 and Table 3.2 perform even better results that specific trained models.

Next experiment that we do is the same like the previous one, but now the domain is "SPORT". Training time for this model was 93.0 seconds in total, but 82.97 seconds spend on optimization. This model and test file, how in previous one is run with 300 abstracts. Table 3.12 shows the outcome of the experiment in coarse grained. From the table we can see that this domain provide a better result that "POLITICS" domain, because here we have less annotated words. But, when compared with base experiment from Table 3.2 and Table 3.4 those experiments perform better results that this one.

Entity	**Precision**	**Recall**	**F1 score**
SPORT	0,9432	0,8839	0,9126
Totals	**0,9432**	**0,8839**	**0,9126**

Table 3.12: Results of "SPORT" base model in coarse grained run with "SPORT" abstracts

Also we train a model in fine grained, with total time of 554.9 seconds, with 543.55 seconds spend on optimization. Table 3.13 show that the result is little bit more better that result with model in coarse grained, but still this result is lower that the results for Table 3.6 and Table 3.8.

Entity	Precision	Recall	F1 score
Athlete	0,9713	0,8366	0,8989
Coach	1,0000	0,7500	0,8571
SportsClub	0,9453	0,9041	0,9242
SportsEvent	1,0000	0,7879	0,8814
SportsLeague	0,9418	0,8958	0,9182
SportsManager	1,0000	0,9615	0,9804
SportsTeam	0,9845	0,8301	0,9007
Totals	**0,9592**	**0,8750**	**0,9152**

Table 3.13: Results of "SPORT" base model in fine grained run with "SPORT" abstracts

After provided 2 experiments with trained models for specific domain, the results shows that training a global model will perform better result than training a domain specific model.

Final experiment that we do with this size of abstracts (300 abstracts) is with "TRANSPORTATION" domain. We needed 58.9 seconds to train the model, from which 50.35 seconds on optimization. Table 3.14 show the experiment outcome in coarse grained, where we can see that this result is lower than results from experiments provided in Table 3.2 and Table 3.5.

Entity	Precision	Recall	F1 score
TRANSPORTATION	0,9583	0,9109	0,9340
Totals	**0,9583**	**0,9109**	**0,9340**

Table 3.14: Results of "TRANSPORTATION" base model in coarse grained run with "TRANSPORTATION" abstracts

Finally we make an experiment in fine grained. Total training time for this kind of model was 702.6 seconds, from which 686.50 seconds spend on optimization. In Table 3.15 we can see the results of provided experiment, where those results are even worse that the experiment with coarse grained model, which in previous two domain, "SPORT" and "POLITICS" was not that case. Also those results are worse than the experiments with a global domain in Table 3.6 and Table 3.9.

Entity	Precision	Recall	F1 score
Aircraft	0,9659	0,8333	0,8947
Automobile	1,0000	0,8000	0,8889
Infrastructure	0,9550	0,9550	0,9550
PublicTransitSystem	0,9662	0,9309	0,9482
Ship	1,0000	0,6429	0,7826
SpaceShuttle	1,0000	0,8333	0,9091
SpaceStation	0,0000	1,0000	0,0000
Train	1,0000	1,0000	1,0000
Totals	**0,9660**	**0,9010**	**0,9324**

Table 3.15: Results of "TRANSPORTATION" base model in fine grained run with "TRANSPORTATION" abstracts

In conclusion with the provided experiments in this section, we can say that training a global model and providing a NER is a better, but a little bit slowest solution than training a domain specific model, except the "POLITICS" domain, where the results was better in domain specific model unlike the experiment with a global domain and test file with "POLITICS" abstracts, but worse than experiment with a global domain tested with abstracts from all 3 domains. Then we wanted to see the impact of fine grained trained models, where in most of the cases this kind of models provide a better results than models trained in coarse grained, except the experiment in "TRANSPORTATION" specific domain where the coarse grained model was better that fine grained model.

After we finish with the main experiment, we were interested about the impact of the size of abstracts that will be used for training models. The following two subsections show the behavior of trained models.

3.3.2 Experiments that has less than 300 abstracts in model

In this subsection we want to see the behavior of models that are trained with less than 300 abstracts. First experiment is trained with 10 abstracts, then we have experiments with 20 abstracts, next experiment with 40 abstracts, and finally experiment with 100 abstracts. The order of abstracts, how we said earlier, is based on PageRank.

Model trained with 10 abstracts to every domain. To retrieve links from DBpedia with SPARQL and prepare data to be able to train models with 10 abstract, our algorithm explain in Section 2.4 takes in total 10.81 minutes, which comparing with main experiment, where we need 2.40 hours, is way more faster to prepare data. Of course this indicates that training models will also be faster than in main experiment.

Coarse grained model.

Description of the experiment. How in the main experiment, also here we start with model trained in coarse grained. To train this kind of model we need 19.6 seconds, from which 17.44 seconds spend on optimization.

Results of the experiment. From Table 3.16 we see that trained model perform the best results without any loosing of words in "SPORT" and "TRANSPORTATION" domains, but worst result in "POLITICS" domain. The result of "POLITICS" domain is even worst than the result from main experiment provided in Table 3.2. Because of this, there is a little bit lower overall result than in the main experiment. This can indicates that training models with lowest number of abstracts, for this kind of domains, is not worth. But let's see how model will behaves when is tested with abstracts from a specific domains.

Entity	**Precision**	**Recall**	**F1 score**
POLITICS	0,9655	0,9333	0,9492
SPORT	1,0000	1,0000	1,0000
TRANSPORTATION	1,0000	1,0000	1,0000
Totals	**0,9811**	**0,9630**	**0,9720**

Table 3.16: Results of global model in coarse grained run with 10 abstracts from every domain

Description of the experiment. In this experiment we take the same model from the previous one, but now tested only with dataset that contains abstracts from "POLITICS" domain.

Results of the experiment. Table 3.17 show the output of experiment where we have global model that is tested with 10 abstracts from "POLITICS" domain. Here model do not annotated any word from other domains unlike in the main experiment in Table 3.3, but even this and the fact that here are much less abstracts does not help to provide a better results.

Entity	**Precision**	**Recall**	**F1 score**
POLITICS	0,9655	0,3636	0,5283
Totals	**0,9655**	**0,3636**	**0,5283**

Table 3.17: Results of global model in coarse grained run with 10 abstracts from "POLITICS" domain

Description of the experiment. For purposes of this experiment we have again used global model, but now tested with abstracts only from "SPORT" domain.

Results of the experiment. Table 3.18 show the outcome of the experiment. We can see that model perform perfect result, how in experiment in Table 3.16 without any misleading annotations, which we cannot say for

the main experiment where model annotated words from ”POLITICS” and ”TRANSPORTATION” domains.

Entity	Precision	Recall	F1 score
SPORT	1,0000	1,0000	1,0000
Totals	**1,0000**	**1,0000**	**1,0000**

Table 3.18: Results of global model in coarse grained run with 10 abstracts from ”SPORT” domain

Description of the result. Finally we tested the model with abstract from ”TRANSPORTATION” domain.

Results of the experiment. From Table 3.19 we can see that model as well as in previous experiment perform maximum result without misleading annotations, unlike the main experiment where how we can see from Table 3.5 model annotate words with ”SPORT” domain and has a lowest result than this one.

Entity	Precision	Recall	F1 score
TRANSPORTATION	1,0000	1,0000	1,0000
Totals	**1,0000**	**1,0000**	**1,0000**

Table 3.19: Results of global model in coarse grained run with 10 abstracts from ”TRANSPORTATION” domain

In conclusion with the results from provided experiments we see that there is a huge impact of the number of abstracts for training a global model in coarse grained. We see that for ”SPORT” and ”TRANSPORTATION” domain our model provide maximum results, which is what we want to reach.

Fine grained model. After we finish the experiments with global models in coarse grained, we wanted to see the impact of fine grained model. Does also here this kind of model will perform better results as was the case in main experiment, where global fine grained model perform a slide better results.

Description of the experiment. Training a fine grained model takes in total 124.7 seconds, from which 120,73 seconds spent in optimization. In this experiment, the model is tested with the same dataset that is trained.

Results of the experiment. From Table 3.20 we can see that now fine grained model provide exactly the same overall result as well as coarse grained model. Also from table we can see in which ontology type our model fails to perform maximum result. So, because of PoliticalParty type where we have a lowest result, the total result is not at the maximum level, even thought other ontology types has maximum annotation.

Entity	Precision	Recall	F1 score
Aircraft	1,0000	1,0000	1,0000
Athlete	1,0000	1,0000	1,0000
Coach	1,0000	1,0000	1,0000
PoliticalParty	0,9600	0,9231	0,9412
Politician	1,0000	1,0000	1,0000
PublicTransitSystem	1,0000	1,0000	1,0000
Ship	1,0000	1,0000	1,0000
SportsClub	1,0000	1,0000	1,0000
SportsEvent	1,0000	1,0000	1,0000
SportsLeague	1,0000	1,0000	1,0000
SportsTeam	1,0000	1,0000	1,0000
Totals	**0,9811**	**0,9630**	**0,9720**

Table 3.20: Results of global model in fine grained run with 10 abstracts from every domain

Description of the experiment. Then how in previous experiments, we take the global train model and test it with abstracts from every specific domain separately.

Results of the experiment. The first domain abstracts was from "POLITICS" domain, where from Table 3.21 we can see that model perform same result as well as in coarse grained model experiment in Table 3.17. Also even our model and test files has words with Election ontology type, the model do not recognize any of them. With that misleading we have lower results, if that doesn't happen the model will perform pretty much good recognition.

Entity	Precision	Recall	F1 score
Election	0,0000	0,0000	0,0000
PoliticalParty	0,9600	0,9231	0,9412
Politician	1,0000	1,0000	1,0000
Totals	**0,9655**	**0,3636**	**0,5283**

Table 3.21: Results of global model in fine grained run with 10 abstracts from "POLITICS" domain

Description of the experiment. In this experiment we also used the global model, but now tested with dataset that contains only abstracts from "SPORT" domain.

Results of the experiment. Table 3.22 shows that our model recognizes all annotated words from test file without any misleading and perform maximum F1 score. In comparing with the main experiment in Table 3.8 where we have some loosing, here that is not the case and it is what we want to reach.

Entity	Precision	Recall	F1 score
Athlete	1,0000	1,0000	1,0000
Coach	1,0000	1,0000	1,0000
SportsClub	1,0000	1,0000	1,0000
SportsEvent	1,0000	1,0000	1,0000
SportsLeague	1,0000	1,0000	1,0000
SportsTeam	1,0000	1,0000	1,0000
Totals	**1,0000**	**1,0000**	**1,0000**

Table 3.22: Results of global model in fine grained run with 10 abstracts from "SPORT" domain

Description of the experiment. Final experiment that we do with global train model was with "TRANSPORTATION" domain abstracts.

Results of the experiment. In Table 3.23 we can see that model, same as in previous experiment with "SPORT" abstracts, perform maximum F1 score result, which in comparing with the main experiment from Table 3.9 here we have improvements on result.

Entity	Precision	Recall	F1 score
Aircraft	1,0000	1,0000	1,0000
PublicTransitSystem	1,0000	1,0000	1,0000
Ship	1,0000	1,0000	1,0000
Totals	**1,0000**	**1,0000**	**1,0000**

Table 3.23: Results of global model in fine grained run with 10 abstracts from "TRANSPORTATION" domain

In conclusion from the provided experiments where we had 10 abstracts on every domain, in comparing with the main experiments, we can say that there is an impact on performing a NER with a smallest number of abstracts for training testing models. Here when we use coarse of fine grained global model and test it with texts from specific domain, except the "POLITICS" domain abstracts, on other two domain, model perform NER without any misleading, which is what we wanted to reach. Also training such small models takes way more less time, than training a big models.

3.3.2.1 Evaluation of domain specific models

In next 6 experiments trained models has 10 domain specific abstracts per model and also test files have the same specification.

Description of the experiment: First domain that we provide an experiment was "POLITICS" specific domain. To train this model we need 3.8 seconds, from which 2.56 seconds spent in optimization. The model is tested with the same dataset that is trained.

Results of the experiment: Table 3.24 show the outcome of the experiment, where the result here is way better, than with comparing with main experiment in Table 3.10 and the experiment with global train model tested with "POLITICS" domain specific text in Table 3.17.

Entity	**Precision**	**Recall**	**F1 score**
POLITICS	0,9737	0,9610	0,9673
Totals	**0,9737**	**0,9610**	**0,9673**

Table 3.24: Results of "POLITICS" domain specific model in coarse grained run with 10 abstracts from the same domain

Description of the experiment: Because we want to know the impact when model is trained in fine grained, we make an experiment with fine grained model. For training this model we need 8.3 seconds, from which 6.99 seconds spent in optimization. As well here the model is tested with the same dataset like is trained.

Results of the experiment: From Table 3.25 we can see that this kind of model provide a higher result than coarse grained model from previous experiment. Also this result is better than result from main experiment in Table 3.11 and the result from the experiment where we tested the global trained model with domain specific text in Table 3.21.

Entity	**Precision**	**Recall**	**F1 score**
Election	1,0000	0,9333	0,9655
PoliticalParty	0,9600	0,9231	0,9412
Politician	1,0000	1,0000	1,0000
Totals	**0,9867**	**0,9610**	**0,9737**

Table 3.25: Results of "POLITICS" domain specific model in fine grained run with 10 abstracts from the same domain

From the "POLITICS" domain specific experiments we can say that for this kind of domain with a lower number of abstracts for training a model the application provide NER with better results unlike the same experiments from the main experiment, where we have a worst results than here.

Description of the experiment: Training for a "SPORT" domain coarse grained model with 10 abstracts we need 5.0 seconds, from which 3.50 seconds spend on optimization. The model is tested with the same dataset with which was trained.

Results of the experiment: From Table 3.26 is clear that this model, same as experiment in Table 3.18, provide excellent result unlike the main experiment in Table 3.12 where we have some loosing in recognition.

Entity	Precision	Recall	F1 score
SPORT	1,0000	1,0000	1,0000
Totals	**1,0000**	**1,0000**	**1,0000**

Table 3.26: Results of "SPORT" domain specific model in coarse grained run with 10 abstracts from the same domain

Description of the experiment: Same as in the previous experiment, also here we have a fine grained model. Time needed for training this model was 26.4 seconds, from which 24.64 seconds spent in optimization. Of course, the model is tested with the dataset with which was trained.

Results of the experiment: From Table 3.27 we see that the result is exactly the same like in coarse grained model (see Table 3.26) and the experiment from Table 3.22 where model is trained with abstracts from every domain and test file contains only abstracts from "SPORT" domain. Those results from Table 3.27 are of course better that the results from the main experiment in Table 3.13, because here we don't have any false recognition.

Entity	Precision	Recall	F1 score
Athlete	1,0000	1,0000	1,0000
Coach	1,0000	1,0000	1,0000
SportsClub	1,0000	1,0000	1,0000
SportsEvent	1,0000	1,0000	1,0000
SportsLeague	1,0000	1,0000	1,0000
SportsTeam	1,0000	1,0000	1,0000
Totals	**1,0000**	**1,0000**	**1,0000**

Table 3.27: Results of "SPORT" domain specific model in fine grained run with 10 abstracts from the same domain

From the experiments provided in Table 3.26 and Table 3.27 as well as in previous experiment the number of abstracts needed for training a model plays significant role also in "SPORT" domain. Here as well there is no difference if model is trained in coarse or fine grain, because we have the same result, but here plays role the time needed for training those models.

Description of the experiment. Finally we have "TRANSPORTATION" domain. For training a coarse grain model we needed 4.3 seconds, from which 3.10 seconds spent in optimization.

Results of the experiment: From Table 3.28 we can see that as well as in "SPORT" specific model NER is provided without any wrong recognition, which was also the case in experiment from Table 3.19. This means that this model also turned out to be better than the main experiment who has 300 abstracts (see Table 3.14).

Entity	Precision	Recall	F1 score
TRANSPORTATION	1,0000	1,0000	1,0000
Totals	**1,0000**	**1,0000**	**1,0000**

Table 3.28: Results of "TRANSPORTATION" domain specific model in coarse grained run with 10 abstracts from the same domain

Description of the experiment. With total time of 14.3 seconds, from which 12.99 seconds spent on optimization we trained a fine grained model for "TRANSPORTATION" domain.

Results of the experiment: Table 3.29 shows that this model 100% precise same as the coarse grain model (see Table 3.29) and the experiment where model is trained with abstracts from every domain and test file contains only abstract from "TRANSPORTATION" domain (see Table 3.23). Of course this experiment provides a better result that the main experiment in Table 3.15.

Entity	Precision	Recall	F1 score
Aircraft	1,0000	1,0000	1,0000
PublicTransitSystem	1,0000	1,0000	1,0000
Ship	1,0000	1,0000	1,0000
Totals	**1,0000**	**1,0000**	**1,0000**

Table 3.29: Results of "TRANSPORTATION" domain specific model in fine grained run with 10 abstracts from the same domain

"TRANSPORTATION" domain model as well as previous two domain models provide a better NER than the main experiment, but of course here we have much less annotated words in model and if we test this model with some other data, the results will be worse than in the main experiment where we have much more data.

3.3.2.2 Evaluation of global domain with 20 abstracts from every domain

Datasets with 20 abstracts for every domain. Since our goal was to know the impact of train data, we decided to increase retrieved abstract from DBpedia to 20 abstracts per domain. Time need to retrieved those abstracts and prepare datasets for training in Stanford NER was 21.30 minutes.

Description of the experiment: In Table 3.30 we provide an experiment where the model was trained with abstracts from every domain, in total 60 abstracts, annotated in coarse grain. We need 36.6 seconds to train the model, from which 32.52 seconds spent in optimization. The model was tested with the same dataset that was trained.

Results of the experiment: Table 3.30 show the output of the experiment, where the overall precision is on maximum level, the recall on "POLITICS" entities is a little bit lower, which results with overall lower recall and not bad at all F1 overall score. For the "SPORT" and "TRANSPORTATION" entities we have a maximum recognition. Referring to the main experiment from Table 3.2 is clearly that results here are better than in main experiment.

Entity	**Precision**	**Recall**	**F1 score**
POLITICS	1,0000	0,9615	0,9804
SPORT	1,0000	1,0000	1,0000
TRANSPORTATION	1,0000	1,0000	1,0000
Totals	**1,0000**	**0,9780**	**0,9889**

Table 3.30: Results of global model in coarse grained run with 20 abstracts from every domain

Description of the experiment: For purposes of the experiment in Table 3.31 we have use the same global trained model from the previous experiment, but now the test file contains only 20 abstracts from "POLITICS" domain.

Results of the experiment: From Table 3.31 we see that the result is way more worst than the previous experiment, but thanks to maximum precision and not recognizing any entity from other domains is slightly better than main experiment from Table 3.3 where model recognize entity from "TRANSPORTATION" domain.

Entity	**Precision**	**Recall**	**F1 score**
POLITICS	1,0000	0,3906	0,5618
Totals	**1,0000**	**0,3906**	**0,5618**

Table 3.31: Results of global model in coarse grained run with 20 abstracts from "POLITICS" domain

Description of the experiment: This experiment is almost identical like previous one, with only difference is test file, where now we tested with abstracts from "SPORT" domain.

Results of the experiment: From Table 3.32 we see that model provide maximum recognition without any wrong entity recognition of other domains. But this is not the case in the main experiment from Table 3.4 where also recognize entities from other two domains, even the file contains only abstracts from "SPORT" domain.

Entity	Precision	Recall	F1 score
SPORT	1,0000	1,0000	1,0000
Totals	**1,0000**	**1,0000**	**1,0000**

Table 3.32: Results of global model in coarse grained run with 20 abstracts from ”SPORT” domain

Description of the experiment: The final experiment is also the same like previous two, where now test file contains only abstracts from ”TRANSPORTATION” domain.

Results of the experiment: Table 3.33 shows that for ”TRANSPORTATION” entities we have maximum recognition, but model also make a wrong entity recognition from ”SPORT” domain. This makes the overall result not to be on his maximum and also in comparing with the main experiment from Table 3.5 where the model also recognize entity from ”SPORT” domain, here the overall result is worst.

Entity	Precision	Recall	F1 score
SPORT	0,0000	1,0000	0,0000
TRANSPORTATION	1,0000	1,0000	1,0000
Totals	**0,9231**	**1,0000**	**0,9600**

Table 3.33: Results of global model in coarse grained run with 20 abstracts from ”TRANSPORTATION” domain

Description of the experiment: Experiment in Table 3.34 is provided with same data like the experiment in Table 3.30, but now the model and test data are annotated in fine grained. We needed in total 239.1 seconds to train model, from which 233.18 seconds spent in optimization.

Results of the experiment: How we can see from Table 3.34 our model provide maximum precision, but because there are 2 ontology types from ”TRANSPORTATION” domain, where out model provide a half on the maximum in the recall we have a lower result at the end. Also in comparing with the main experiment from Table 3.6 we have slightly lower results here. As well those results are lower than the experiment in coarse grain (see Table 3.30).

Entity	Precision	Recall	F1 score
Aircraft	1,0000	0,5000	0,6667
Athlete	1,0000	1,0000	1,0000
Coach	1,0000	1,0000	1,0000
Infrastructure	1,0000	0,5000	0,6667
PoliticalParty	1,0000	0,9512	0,9750
Politician	1,0000	1,0000	1,0000
Ship	1,0000	1,0000	1,0000
SpaceShuttle	1,0000	1,0000	1,0000
SpaceStation	1,0000	1,0000	1,0000
SportsClub	1,0000	1,0000	1,0000
SportsEvent	1,0000	1,0000	1,0000
SportsLeague	1,0000	1,0000	1,0000
SportsTeam	1,0000	1,0000	1,0000
Totals	**1,0000**	**0,9560**	**0,9775**

Table 3.34: Results of global model in fine grained run with 20 abstracts from every domain

Description of the experiment: In this experiment we use the same trained model from previous experiment, but now the test file contains only abstracts from ”POLITICS” domain, who is annotated in fine grain.

Results of the experiment: Table 3.35 show the output of the experiment, where we can see that even we have Election type on model and test file, the model do not find any entity with that type. Also for the Politician type we have a very low recall, which reflects that there is a very low overall result. In comparing with the experiment in coarse grain (see Table 3.31 we have exactly the same overall result, but when we compare with the main experiment from Table 3.7 even in that experiment model also annotate some words from other two domains, the overall result is better than the result here.

Entity	Precision	Recall	F1 score
Election	0,0000	0,0000	0,0000
PoliticalParty	1,0000	0,9512	0,9750
Politician	1,0000	0,2000	0,3333
Totals	**1,0000**	**0,3906**	**0,5618**

Table 3.35: Results of global model in fine grained run with 20 abstracts from ”POLITICS” domain

Description of the experiment: How in the previous experiment also here we have the same model but now tested with abstracts from ”SPORT” domain annotated in fine grain.

Results of the experiment: In Table 3.36 we have the output of the provided experiment. How we can see the results are excellent, there is no wrong recognition or some lower values of precision and recall. When comparing with experiment in coarse grain (see Table 3.32) we have the same overall result, but we cannot say that about the results from the main experiment provided in Table 3.8 where we have wrong recognition of entities from other 2 domains and only one type has maximum precision and recall.

Entity	**Precision**	**Recall**	**F1 score**
Athlete	1,0000	1,0000	1,0000
Coach	1,0000	1,0000	1,0000
SportsClub	1,0000	1,0000	1,0000
SportsEvent	1,0000	1,0000	1,0000
SportsLeague	1,0000	1,0000	1,0000
SportsTeam	1,0000	1,0000	1,0000
Totals	**1,0000**	**1,0000**	**1,0000**

Table 3.36: Results of global model in fine grained run with 20 abstracts from "SPORT" domain

Description of the experiment: The last experiment with the model used in previous 3 experiment is now the dataset test file with abstracts from "TRANSPORTATION" domain also annotated in fine grain.

Results of the experiment: Table 3.37 shows that our train model provide the same results how in experiment in Table 3.34 for the "TRANSPORTATION" ontology types. Also we have exactly the same overall result with the experiment in coarse grain (see Table 3.33, but when we compare results with the main experiment from Table 3.9 we have a way more better results than here, even thought that there model recognize some entities from other two domains.

Entity	**Precision**	**Recall**	**F1 score**
Aircraft	1,0000	0,5000	0,6667
Infrastructure	1,0000	0,5000	0,6667
Ship	1,0000	1,0000	1,0000
SpaceShuttle	1,0000	1,0000	1,0000
SpaceStation	1,0000	1,0000	1,0000
SportsTeam	0,0000	1,0000	0,0000
Totals	**0,9091**	**0,8333**	**0,8696**

Table 3.37: Results of global model in fine grained run with 20 abstracts from "TRANSPORTATION" domain

3.3.2.3 Evaluation of domain specific models with 20 abstracts

Description of the experiment: Experiment in Table 3.38 was provided with model trained only with abstracts from "POLITICS" domain in coarse grain. To train this model we needed 5.4 seconds, from which 3.81 seconds spent in optimization.

Results of the experiment: How we can see from Table 3.38 the result is not bad at all. In comparing with experiment from Table 3.31 the results now are way more better and are more usable. Also referencing to main experiment from Table 3.10 where the only difference is the number of abstracts used for training the model, now the result is a little bit better than there.

Entity	**Precision**	**Recall**	**F1 score**
POLITICS	0,9921	0,9766	0,9843
Totals	**0,9921**	**0,9766**	**0,9843**

Table 3.38: Results of "POLITICS" domain specific model in coarse grained run with 20 abstracts from the same domain

Description of the experiment: For the purposes of this experiment we have used the same data from the previous one, but now annotated in fine grain. To train this kind of model we needed 12.4 seconds, from which 10.80 seconds spent in optimization.

Results of the experiment: We tested the model with the same data that was created and how we can see from Table 3.39 model provides maximum precision on entities, but because of lower recall we have overall a quite lower F1 score. But in comparing with previous experiment the result is slightly better, which we cannot say that about experiment in Table 3.35 where result is terrible. Also in comparing with the main experiment from Table 3.11 now model provides also a little bit better result, but not that significant like in experiment from Table 3.35

Entity	**Precision**	**Recall**	**F1 score**
Election	1,0000	0,9688	0,9841
PoliticalParty	1,0000	0,9512	0,9750
Politician	1,0000	1,0000	1,0000
Totals	**1,0000**	**0,9766**	**0,9881**

Table 3.39: Results of "POLITICS" domain specific model in fine grained run with 20 abstracts from the same domain

Description of the experiment: Experiment in Table 3.40 is provided with a coarse grain model trained with abstracts only from "SPORT" domain. The time needed to train this model was 5.0 seconds, from which 3.50 seconds

spent in optimization. To test it we have used the same dataset that model was trained.

Results of the experiment: How we can see from Table 3.40 the trained model provide excellent recognition on entities from test dataset, without any miss or wrong recognition. The same results we have in experiment with a global domain model tested with the same dataset (see Table 3.32). But when we compare the results from here and the results from the main experiment (see Table 3.12) we see that now results are better, but here we have a less entities.

Entity	**Precision**	**Recall**	**F1 score**
SPORT	1,0000	1,0000	1,0000
Totals	**1,0000**	**1,0000**	**1,0000**

Table 3.40: Results of "SPORT" domain specific model in coarse grained run with 20 abstracts from the same domain

Description of the experiment: In this experiment we have used the same dataset from previous, but now entities are annotated in fine grain. To train a fine grain model we have need 26.4 seconds, from which 24.46 second spent in optimization. As well as previous the model is tested with the dataset that is trained.

Results of the experiment: In Table 3.41 we see the output of the experiment. How in the previous experiment in coarse grain, also here the results are excellent without any looseness of unrecognized entities. As well we have the same result in Table 3.36 where we have a global domain and the same test file (test file contains only abstracts from "SPORT" domain). But in comparing with the main experiment from Table 3.13, again the results there are lower than here (experiment from Table 3.41).

Entity	**Precision**	**Recall**	**F1 score**
Athlete	1,0000	1,0000	1,0000
Coach	1,0000	1,0000	1,0000
SportsClub	1,0000	1,0000	1,0000
SportsEvent	1,0000	1,0000	1,0000
SportsLeague	1,0000	1,0000	1,0000
SportsTeam	1,0000	1,0000	1,0000
Totals	**1,0000**	**1,0000**	**1,0000**

Table 3.41: Results of "SPORT" domain specific model in fine grained run with 20 abstracts from the same domain

Description of the experiment: At the end we train a model with abstracts from "TRANSPORTATION" domain. To train a coarse grain model from this domain we needed 4.3 seconds, from which 3.10 seconds spent in

optimization. Of course, here also we tested the model with the same dataset that was created.

Results of the experiment: In Table 3.42 we see that the model provide maximum precision, but lower recall on entities which results with a worst F1 score. Surprisingly result here is lower than the experiment where we had a global model tested with the same dataset like here (see Table 3.33), even those that there we have a wrong entity recognition from "SPORT" domain, the results is still better. As well the results from the main experiment in Table 3.14 are better than here, which was not the case in the previous 4 experiments.

Entity	**Precision**	**Recall**	**F1 score**
TRANSPORTATION	1,0000	0,8333	0,9091
Totals	**1,0000**	**0,8333**	**0,9091**

Table 3.42: Results of "TRANSPORTATION" domain specific model in coarse grained run with 20 abstracts from the same domain

Description of the experiment: We also train a fine grain model from "TRANSPORTATION" domain. To train it we needed 14.3 seconds, from which 12.99 seconds spent in optimization. As well as the previous experiment, the model is tested with the same dataset that is created.

Results of the experiment: The output of the experiment seen in Table 3.43 are quite surprisingly. Until now in the experiments with a lower abstracts than the main experiment, the fine grained models provides same or better results than coarse grained model. Here model provides a worst result than the previous experiment. Also experiment in Table 3.37 and the main experiment from Table 3.15 have better results than here.

Entity	**Precision**	**Recall**	**F1 score**
Aircraft	1,0000	0,5000	0,6667
Infrastructure	1,0000	0,5000	0,6667
Ship	1,0000	1,0000	1,0000
SpaceShuttle	1,0000	1,0000	1,0000
SpaceStation	1,0000	1,0000	1,0000
Totals	**1,0000**	**0,7500**	**0,8571**

Table 3.43: Results of "TRANSPORTATION" domain specific model in fine grained run with 20 abstracts from the same domain

Until now we can say that the number of abstracts used to train a models has an impact on the final results and also it is faster to train a smaller models, but in that case we are short on entities and types. We still have 2 more groups of different number of abstracts, 40 abstracts per domain and 100 abstracts

per domain. So let see how those groups will behave in comparing with the main experiment, where we have 300 abstracts per domain.

3.3.2.4 Evaluation of global domain with 40 abstracts from every domain

Datasets with 40 abstracts for every domain. Now we have increased number of retrieved links from DBpedia to 40 abstracts. To retrieve links and prepare datasets that contains 40 abstracts to every domain our algorithm needs 30.94 minutes.

Description of the experiment: In Table 3.44 we provide an experiment where the model is trained with all retrieved abstracts (120 abstracts in total) in coarse grain. To train this model with Stanford NER we needed 101.7 seconds, from which 91.52 seconds spent in optimization. We have tested the model with the same dataset that was created.

Results of the experiment: Table 3.44 shows the output of the experiment, where we see that for the "SPORT" domain we have maximum results, but also results from other domains are not bad at all. This gives a very good total results on precision, recall and F1 score. In comparing with the main experiment from Table 3.2 now we have a little bit more better results, but with a lower number of entities in model.

Entity	**Precision**	**Recall**	**F1 score**
POLITICS	0,9890	0,9375	0,9626
SPORT	1,0000	1,0000	1,0000
TRANSPORTATION	1,0000	0,9846	0,9922
Totals	**0,9960**	**0,9724**	**0,9841**

Table 3.44: Results of global model in coarse grained run with 40 abstracts from every domain

Description of the experiment: In this experiment we have used the same model from previous, but now it is tested with dataset that contains only abstracts from "POLITICS" domain.

Results of the experiment: How we see from Table 3.45 even those that we don't have any recognition from other domains and a maximum precision, the recall is very low which reflects with low F1 score. When we compare with the previous experiment we see that there result for "POLITICS" domain is better than here. Also in comparing with the main experiment from Table 3.3 where model recognize some entities from "TRANSPORTATION" domain and precision is lower, still those results are better than here.

Entity	Precision	Recall	F1 score
POLITICS	0,9890	0,3529	0,5202
Totals	**0,9890**	**0,3529**	**0,5202**

Table 3.45: Results of global model in coarse grained run with 40 abstracts from "POLITICS" domain

Description of the experiment: For purposes of this experiment we also used the global modal, but now it is tested with dataset that contains only abstracts from "SPORT" domain.

Results of the experiment: How we can see from Table 3.46 model provides perfect recognition without any wrong entity recognition from other domains. The same result for "SPORT" domain we have in experiment from Table 3.44, but in comparing with the main experiment from Table 3.4 where model recognize some entities from other two domain, as well the result for "SPORT" domain are lower than here.

Entity	Precision	Recall	F1 score
SPORT	1,0000	1,0000	1,0000
Totals	**1,0000**	**1,0000**	**1,0000**

Table 3.46: Results of global model in coarse grained run with 40 abstracts from "SPORT" domain

Description of the experiment: Finally we tested the global domain with abstracts from "TRANSPORTATION" domain.

Results of the experiment: As we can see from Table 3.47 model provides maximum precision on "TRANSPORTATION" domain, but a little bit lower recall, which of course reflects on F1 score. Model also recognize some wrong entities from "SPORT" domain which results with lower total F1 score. As well here we have same results like in experiment from Table 3.44. In comparing with the main experiment from Table 3.5 where model also recognizes wrong entities from "SPORT" domain the overall result is slightly better than here.

Entity	Precision	Recall	F1 score
SPORT	0,0000	1,0000	0,0000
TRANSPORTATION	1,0000	0,9846	0,9922
Totals	**0,9697**	**0,9846**	**0,9771**

Table 3.47: Results of global model in coarse grained run with 40 abstracts from "TRANSPORTATION" domain

Description of the experiment: For the purposes of this experiment we train a fine grain model with abstracts from every domain. The time that we

needed to train this model was 287.5 second, from which 278.99 seconds spent in optimization. The model is tested with the same dataset that is created.

Results of the experiment: How we can see from Table 3.48 the list of ontology types now is longer than in previous two groups of experiment (with 10 and 20 abstracts). Also model provides maximum precision on every type except PoliticalParty type and a lower recall on the same type as well the Politician ontology type. This results with a lower overall result on every measurement. But in comparing with the model in coarse grain (see Table 3.44), here the overall result is better, but not what significant. In comparing with the main experiment from Table 3.6 where model has more ontology types, the overall result is lower than now, but the difference in the results is not that big. Another thing is that even test dataset contains Election type, model do not find any entity of that type.

Entity	**Precision**	**Recall**	**F1 score**
Aircraft	1,0000	1,0000	1,0000
Athlete	1,0000	1,0000	1,0000
Coach	1,0000	1,0000	1,0000
Infrastructure	1,0000	1,0000	1,0000
PoliticalParty	0,9863	0,9730	0,9796
Politician	1,0000	0,8182	0,9000
PublicTransitSystem	1,0000	1,0000	1,0000
Ship	1,0000	1,0000	1,0000
SpaceShuttle	1,0000	1,0000	1,0000
SpaceStation	1,0000	1,0000	1,0000
SportsClub	1,0000	1,0000	1,0000
SportsEvent	1,0000	1,0000	1,0000
SportsLeague	1,0000	1,0000	1,0000
SportsTeam	1,0000	1,0000	1,0000
Totals	**0,9960**	**0,9764**	**0,9861**

Table 3.48: Results of global model in fine grained run with 40 abstracts from every domain

Description of the experiment: In this experiment we have used the same model from previous, but now it is tested with the dataset that contains only abstracts from ”POLITICS” domain, of course annotated in fine grain.

Results of the experiment: From Table 3.49 we see that model almost fail the test, even thought that test data are part of training the model. In test dataset we have entities with Election ontology type, but model do not recognize any of them, as well on Politician type we have maximum precision and a very low recall which reflects with low F1 score. Those results are the same with the experiment in coarse grain from Table 3.45. As we compare the results for every ontology type from previous experiment we will get way more

better results, also the results from the main experiment in Table 3.7 are a slightly better although model recognize types of entities from other domains.

Entity	Precision	Recall	F1 score
Election	0,0000	0,0000	0,0000
PoliticalParty	0,9863	0,9730	0,9796
Politician	1,0000	0,1565	0,2707
Totals	**0,9890**	**0,3529**	**0,5202**

Table 3.49: Results of global model in coarse grained run with 40 abstracts from "POLITICS" domain

Description of the experiment: Experiment here has the same trained model like the previous one, with the difference that now it is tested with dataset that contains abstracts only from "SPORT" domain.

Results of the experiment: The output of the experiment shown in Table 3.50 is exactly that we wanted to reach. Model provide maximum results on every ontology type without any wrong recognition. The same results for "SPORT" ontology types we have in experiment with the global dataset in Table 3.48 and the experiment in coarse grain from Table 3.46. We cannot say that about the main experiment in Table 3.8 where some ontology types don't have maximum results and also model recognize entities from other two domains, which results with lower total result than here.

Entity	Precision	Recall	F1 score
Athlete	1,0000	1,0000	1,0000
Coach	1,0000	1,0000	1,0000
SportsClub	1,0000	1,0000	1,0000
SportsEvent	1,0000	1,0000	1,0000
SportsLeague	1,0000	1,0000	1,0000
SportsTeam	1,0000	1,0000	1,0000
Totals	**1,0000**	**1,0000**	**1,0000**

Table 3.50: Results of global model in coarse grained run with 40 abstracts from "SPORT" domain

Description of the experiment: The final experiment with the fine grain global model that we provide was that now it is tested with the abstracts from "TRANSPORTATION" domain.

Results of the experiment: As we can see from Table 3.51, model provide maximum results for ontology types from "TRANSPORTATION" domain, but also recognize some entities from "SPORT" domain, which was not in test dataset. Because of that the total result is a little bit lower that the maximum. When we compare the results for every type with the results from

experiment in Table 3.48 we can see that are the same. Also this experiment provides better result that the coarse grain experiment from Table 3.47. A little bit surprisingly is that, that the main experiment from Table 3.9 gives better results than the experiment here.

Entity	Precision	Recall	F1 score
Aircraft	1,0000	1,0000	1,0000
Infrastructure	1,0000	1,0000	1,0000
PublicTransitSystem	1,0000	1,0000	1,0000
Ship	1,0000	1,0000	1,0000
SpaceShuttle	1,0000	1,0000	1,0000
SpaceStation	1,0000	1,0000	1,0000
SportsTeam	0,0000	1,0000	0,0000
Totals	**0,9701**	**1,0000**	**0,9848**

Table 3.51: Results of global model in coarse grained run with 40 abstracts from ”TRANSPORTATION” domain

3.3.2.5 Evaluation of domain specific models with 40 abstracts

Description of the experiment: The experiment here is provided with coarse grain model that is trained with dataset that contains only abstracts from ”POLITICS” domain. To train this model we needed 13.9 seconds, from which 10.36 seconds spent in optimization. It is tested with the same dataset that is trained.

Results of the experiment: Table 3.52 shows the outcome of the experiment, where in comparing with experiment from Table 3.44 we can see a big improvement on the result, where now we are closer to the best performance. In comparing with the main experiment from Table 3.10, now we have a way more better result than there.

Entity	Precision	Recall	F1 score
POLITICS	0,9921	0,9804	0,9862
Totals	**0,9921**	**0,9804**	**0,9862**

Table 3.52: Results of ”POLITICS” domain specific model in coarse grained run with 40 abstracts from the same domain

Description of the experiment: This experiment is provided with same data like the previous one, but now dataset is annotated in fine grain. To train a ”POLITICS” fine grain model we needed 41.0 seconds, from which 36.11 seconds spent in optimization. Of course it is tested with the dataset that is trained.

Results of the experiment: As we can see from Table 3.53 the result is better than the previous experiment, and also model recognize all entities

from the domain, which was not the case in experiment from Table 3.49 and as well the result now is way more better. Comparing with the main experiment from Table 3.11, now we again have a better results to every ontology type, and of course better total result.

Entity	Precision	Recall	F1 score
Election	1,0000	0,9848	0,9924
PoliticalParty	0,9863	0,9730	0,9796
Politician	1,0000	1,0000	1,0000
Totals	**0,9960**	**0,9882**	**0,9921**

Table 3.53: Results of "POLITICS" domain specific model in fine grained run with 40 abstracts from the same domain

Description of the experiment: For the purposes of this experiment we have used dataset with abstracts from "SPORT" domain annotated in coarse grain. To train a coarse grain model we needed 10.0 seconds, from which 6.84 seconds spent in optimization. Here as well we test the model with the dataset that is trained.

Results of the experiment: Table 3.55 show the output of the experiment, where we have a maximum entity recognition, which was also the case in experiment with a global model and same test dataset like here (see Table 3.46. In comparing with the same experiment in Table 3.12 from main experiment, is clearly that now we had a way more better results than there.

Entity	Precision	Recall	F1 score
SPORT	1,0000	1,0000	1,0000
Totals	**1,0000**	**1,0000**	**1,0000**

Table 3.54: Results of "SPORT" domain specific model in coarse grained run with 40 abstracts from the same domain

Description of the experiment: Then we used same data like in previous experiment, but now dataset is annotated in fine grain. To train a "SPORT" fine grain domain model with Stanford NER we needed 56.2 seconds, from which 52.77 seconds spent in optimization. It is tested as always, with the same dataset that is trained.

Results of the experiment: In Table 3.55 we see the experiment outcome, where because of the lower recall on SportsClub ontology type, the overall result is slightly lower. So after some time, again fine grain model provides a little bit worst result than coarse grain model. Because of this the experiment with a global model and same test dataset like here from Table 3.50 gives better results. For consolation is the fact that model here provide significantly better result than the main experiment in Table 3.13

Entity	Precision	Recall	F1 score
Athlete	1,0000	1,0000	1,0000
Coach	1,0000	1,0000	1,0000
SportsClub	1,0000	0,9474	0,9730
SportsEvent	1,0000	1,0000	1,0000
SportsLeague	1,0000	1,0000	1,0000
SportsTeam	1,0000	1,0000	1,0000
Totals	**1,0000**	**0,9890**	**0,9945**

Table 3.55: Results of "SPORT" domain specific model in fine grained run with 40 abstracts from the same domain

Description of the experiment: The final experiments with this number of abstracts is with "TRANSPORTATION" domain. To train a coarse grain model we needed 9.1 seconds, from which 6.24 seconds spent in optimization. As in all previous experiment, test is provided with the same dataset that is trained.

Results of the experiment: Table 3.57 shows the output of the coarse grain model experiment, where the result is very close to the maximum. In experiment where we use a global model who is tested with same dataset as here (see Table 3.47), overall results there is lower, because model recognize some entities from another domain, but the result from "TRANSPORTATION" domain is the same as here. As well this experiment provides a better result than the main experiment from Table 3.14.

Entity	Precision	Recall	F1 score
TRANSPORTATION	1,0000	0,9846	0,9922
Totals	**1,0000**	**0,9846**	**0,9922**

Table 3.56: Results of "TRANSPORTATION" domain specific model in coarse grained run with 40 abstracts from the same domain

Description of the experiment: Lastly we train a fine grain model with the same data like in previous experiment, but of course annotated in fine grain. To train this kind of model we needed 44.6 seconds, from which 41.63 seconds spent in optimization. It is tested as usual.

Results of the experiment: As we can see from Table 3.57 we have exactly the same result like in coarse grain experiment. In comparing with the experiment in Table 3.51 where because of wrong entity recognition model gives lower results than here, although the results on every ontology type is the same, except the PublicTransitSystem type, where now we had a lower recall. Also model here, again gives us better results than the main experiment in Table 3.15.

Entity	Precision	Recall	F1 score
Aircraft	1,0000	1,0000	1,0000
Infrastructure	1,0000	1,0000	1,0000
PublicTransitSystem	1,0000	0,9630	0,9811
Ship	1,0000	1,0000	1,0000
SpaceShuttle	1,0000	1,0000	1,0000
SpaceStation	1,0000	1,0000	1,0000
Totals	**1,0000**	**0,9846**	**0,9922**

Table 3.57: Results of "TRANSPORTATION" domain specific model in fine grained run with 40 abstracts from the same domain

3.3.2.6 Evaluation of global domain with 100 abstracts from every domain

Datasets with 100 abstracts for every domain. In this final group of 100 abstracts per domain we will repeat all experiments like in previous groups, to see how model will behaves when is closer to the number of abstracts in main experiment. To retrieve and prepare datasets for training in Stanford NER, our algorithm needs in total 63.71 minutes, which is twice more than when we had 40 abstracts for every domain.

Description of the experiment: In Table 3.58 we provide an experiment where the model is trained with, in total, 300 abstracts annotated in coarse grain. To train this kind of model we needed 258.1 seconds, from which 246.28 seconds spent in optimization. The model is tested with the dataset that was trained.

Results of the experiment: As we can see from Table 3.58 now model provides maximum result only in precision measurement on "TRANSPORTATION" domain. From here we can see that results are closer to the main experiment from Table 3.2, but still results here are little bit better than there.

Entity	Precision	Recall	F1 score
POLITICS	0,9920	0,9612	0,9764
SPORT	0,9963	0,9926	0,9944
TRANSPORTATION	1,0000	0,9735	0,9865
Totals	**0,9952**	**0,9766**	**0,9856**

Table 3.58: Results of global model in coarse grained run with 100 abstracts from every domain

Description of the experiment: For purposes of this experiment we have used the model train previously, but now it is tested with the dataset

that contains only abstracts from ”POLITICS” domain, annotated in coarse grain.

Results of the experiment: From Table 3.59 we see that even we have a precision close to maximum value, but because of very low recall, the F1 score is lower, which means that model recognize only half of our entities. When we compare with the previous experiment is clearly that there the results for ”POLITICS” domain are better. Also comparably with the main experiment from Table 3.3 results there a quite better than now, although there model recognize some entities from ”TRANSPORTATION” domain.

Entity	**Precision**	**Recall**	**F1 score**
POLITICS	0,9920	0,3615	0,5299
Totals	**0,9920**	**0,3615**	**0,5299**

Table 3.59: Results of global model in coarse grained run with 100 abstracts from ”POLITICS” domain

Description of the experiment: In this experiment we also used the global trained model, but now it is tested with ”SPORT” domain abstracts dataset.

Results of the experiment: Table 3.60 shows the output of the experiment, from where we see that even the results aren't at their maximum are quite satisfying, but are a little bit lower than in global experiment in Table 3.58. As well in comparing with the main experiment from Table 3.4, now, very importantly, we don't have any wrong entity recognition and even if we compare only results from ”SPORT” domain, still result is better now.

Entity	**Precision**	**Recall**	**F1 score**
SPORT	0,9962	0,9888	0,9925
Totals	**0,9962**	**0,9888**	**0,9925**

Table 3.60: Results of global model in coarse grained run with 100 abstracts from ”SPORT” domain

Description of the experiment: Final experiment with the global domain is that is tested with the dataset that contains only abstracts from ”TRANSPORTATION” domain.

Results of the experiment: From Table 3.61 we see that model also recognize some entities from ”SPORT” domain, that are not part of the tested dataset. Also if we compare only results for ”TRANSPORTATION” domain with the results from global experiment in Table 3.58, they are the same, but now because of that wrong recognition, overall results is slightly lower. The exact situation is in main experiment from Table 3.5 where model also make a mistake, but there results are better than here.

Entity	Precision	Recall	F1 score
SPORT	0,0000	1,0000	0,0000
TRANSPORTATION	1,0000	0,9735	0,9865
Totals	**0,9821**	**0,9735**	**0,9778**

Table 3.61: Results of global model in coarse grained run with 100 abstracts from "TRANSPORTATION" domain

Description of the experiment: For purposes of this and the next 3 experiments we train a fine grain global model with 100 abstracts from every domain. To train this kind of model we needed 857.3 seconds, from which 844.91 seconds spent in optimization. In this experiment we tested the model with the dataset that was trained.

Results of the experiment: From Table 3.62 we can see the results for every particular ontology type. Now model preforms bad results mostly for types from "SPORT" domain. And it happens again that the fine grain model gives worst total results than the coarse grain model (see Table 3.58 for coarse grain results). As well referencing to main experiment in Table 3.6 the results there are way more better than here, although there we have more data.

Entity	Precision	Recall	F1 score
Aircraft	1,0000	0,6957	0,8205
Athlete	1,0000	0,4167	0,5882
Automobile	1,0000	1,0000	1,0000
Coach	1,0000	0,6667	0,8000
Infrastructure	1,0000	1,0000	1,0000
PoliticalParty	0,8774	0,6700	0,7598
Politician	1,0000	0,7455	0,8542
PublicTransitSystem	0,9744	0,7308	0,8352
Ship	1,0000	0,6000	0,7500
SpaceShuttle	1,0000	1,0000	1,0000
SpaceStation	1,0000	1,0000	1,0000
SportsClub	0,9512	0,9398	0,9455
SportsEvent	0,9737	0,8605	0,9136
SportsLeague	0,9500	0,8636	0,9048
SportsManager	1,0000	1,0000	1,0000
SportsTeam	1,0000	0,6364	0,7778
Train	1,0000	1,0000	1,0000
Totals	**0,9452**	**0,7535**	**0,8385**

Table 3.62: Results of global model in fine grained run with 100 abstracts from every domain

Description of the experiment: In this experiment we tested the global

model with dataset that contains only abstracts from ”POLITICS” domain, of course annotated in fine grain.

Results of the experiment: Table 3.63 shows the output of the experiment, where model provides terrible results and also recognize entity with SportEvent type which is not part of the dataset, but don't recognize any entity with Election type who is part of the dataset. As well this experiment gives a worst total result than the coarse grain experiment in Table 3.59 and worst type results than global model from previous experiment. Finally the results now are worse than in main experiment from Table 3.7.

Entity	**Precision**	**Recall**	**F1 score**
Election	0,0000	0,0000	0,0000
PoliticalParty	0,8774	0,6700	0,7598
Politician	1,0000	0,1285	0,2278
SportsEvent	0,0000	1,0000	0,0000
Totals	**0,8985**	**0,2580**	**0,4009**

Table 3.63: Results of global model in fine grained run with 100 abstracts from ”POLITICS” domain

Description of the experiment: Here we also have the global train model, but for this experiment is tested with dataset abstracts from ”SPORT” domain.

Results of the experiment: As we see from Table 3.64 model gives exact the same results for every ontology type like in global experiment in Table 3.62, where because of worst recall the overall result are lower than in comparing with the coarse grain experiment in Table 3.60. As well referencing to main experiment in Table 3.8, now again results are worse than there.

Entity	**Precision**	**Recall**	**F1 score**
Athlete	1,0000	0,4167	0,5882
Coach	1,0000	0,6667	0,8000
SportsClub	0,9506	0,9390	0,9448
SportsEvent	1,0000	0,8605	0,9250
SportsLeague	0,9500	0,8636	0,9048
SportsManager	1,0000	1,0000	1,0000
SportsTeam	1,0000	0,6250	0,7692
Totals	**0,9683**	**0,7985**	**0,8753**

Table 3.64: Results of global model in fine grained run with 100 abstracts from ”SPORT” domain

Description of the experiment: The final experiment with the global model is with dataset that contains only abstracts from ”TRANSPORTATION” domain.

Results of the experiment: Table 3.65 shows the outcome of the experiment, where again the results for every individual ontology type aren't increased from the result in global experiment in Table 3.62. Because of worst recall in Aircraft, PublicTransitSystem and Ship types we have a lower total result than in coarse grain experiment in Table 3.61. Not surprisingly for this group of experiment, the results from here are also worst than in main experiment from Table 3.9.

Entity	Precision	Recall	F1 score
Aircraft	1,0000	0,6957	0,8205
Automobile	1,0000	1,0000	1,0000
Infrastructure	1,0000	1,0000	1,0000
PublicTransitSystem	0,9744	0,7308	0,8352
Ship	1,0000	0,6000	0,7500
SpaceShuttle	1,0000	1,0000	1,0000
SpaceStation	1,0000	1,0000	1,0000
SportsClub	0,0000	1,0000	0,0000
SportsTeam	0,0000	1,0000	0,0000
Train	1,0000	1,0000	1,0000
Totals	**0,9677**	**0,7965**	**0,8738**

Table 3.65: Results of global model in fine grained run with 100 abstracts from "TRANSPORTATION" domain

3.3.2.7 Evaluation of domain specific models with 100 abstracts

Description of the experiment: In this experiment we train a domain specific coarse grain model with dataset that contains only abstracts from "POLITICS" domain. The time needed to train this model was 34.2 seconds, from which 28.45 seconds spent in optimization. The model is tested with the same dataset that is trained.

Results of the experiment: As we can see from Table 3.66 this model provides a better result than the 2 experiments with the global model in Table 3.58 and and a way more better results than experiment from Table 3.59. As well referencing to main experiment in Table 3.10, now model gives better results.

Entity	Precision	Recall	F1 score
POLITICS	0,9956	0,9898	0,9927
Totals	**0,9956**	**0,9898**	**0,9927**

Table 3.66: Results of "POLITICS" domain specific model in coarse grained run with 100 abstracts from the same domain

Description of the experiment: With the same data from the previous experiment, but not annotated in fine grain, we train new model. We needed 89.4 seconds to train this model, from which 83.54 seconds spent in optimization. The model is tested with the same dataset that was trained.

Results of the experiment: Table 3.67 shows the outcome of the experiment, where we see that even results on every type are close to maximum, the overall results is slightly lower than the previous experiment. But in comparing to the experiments with global domain in Table 3.62 and Table 3.63, now results are better, we don't have any wrong recognition and also entities from Election ontology type are recognized. Referencing to the main experiment in Table 3.11, now as well results are way more better and more usable.

Entity	**Precision**	**Recall**	**F1 score**
Election	1,0000	0,9878	0,9939
PoliticalParty	0,9950	0,9852	0,9901
Politician	0,9937	0,9906	0,9922
Totals	**0,9956**	**0,9883**	**0,9920**

Table 3.67: Results of "POLITICS" domain specific model in fine grained run with 100 abstracts from the same domain

Description of the experiment: For the purposes of the experiment we train a coarse grain model with dataset that contains only abstracts from "SPORT" domain. Time needed to train this kind of model was 25.3 seconds, from which 20.47 seconds spent in optimization. The model is tested like in previous experiments, with the same dataset that is trained.

Results of the experiment: From Table 3.68 we see that model gives a slightly better results than in experiments with global domain in Table 3.58 and Table 3.60. Also referencing to the main experiment in Table 3.12, now model gives results that are closer to the maximum values, but we don't have that much data here.

Entity	**Precision**	**Recall**	**F1 score**
SPORT	0,9963	0,9963	0,9963
Totals	**0,9963**	**0,9963**	**0,9963**

Table 3.68: Results of "SPORT" domain specific model in coarse grained run with 100 abstracts from the same domain

Description of the experiment: This experiment is provided with a "SPORT" fine grain model. To train this model we needed 194.2 seconds, from which 187.77 seconds spent in optimization. Test is the same like previous experiment, where dataset now are annotated in fine grain.

Results of the experiment: As we see from Table 3.64 the overall results is lower than previous experiment, but way better than experiments provided

with the global model in Table 3.62 and Table 3.64. Also referencing to main experiment in Table 3.13, now model was more precise than there, an because of that gives better results.

Entity	Precision	Recall	F1 score
Athlete	1,0000	0,9722	0,9859
Coach	1,0000	1,0000	1,0000
SportsClub	1,0000	0,9878	0,9939
SportsEvent	1,0000	0,9767	0,9882
SportsLeague	1,0000	1,0000	1,0000
SportsManager	1,0000	1,0000	1,0000
SportsTeam	1,0000	1,0000	1,0000
Totals	**1,0000**	**0,9888**	**0,9944**

Table 3.69: Results of "SPORT" domain specific model in fine grained run with 100 abstracts from the same domain

Description of the experiment: Experiment in Table 3.70 is provided with a coarse grain model train with dataset from "TRANSPORTATION" domain. To train this model we needed 20.5 seconds in total, from which 14.93 seconds spent in optimization. As previous model is tested with same dataset that is trained.

Results of the experiment: Table 3.70 shows that model provides maximum precision on entities, but slight lower recall which results with lower F1 score. Comparably with experiments with global model in Table 3.58 and Table 3.61, now model gives better score than there and it's faster to train and provide experiment. As well referencing to main experiment in Table 3.14 again those results are lower than domain specific model.

Entity	Precision	Recall	F1 score
TRANSPORTATION	1,0000	0,9912	0,9956
Totals	**1,0000**	**0,9912**	**0,9956**

Table 3.70: Results of "TRANSPORTATION" domain specific model in coarse grained run with 100 abstracts from the same domain

Description of the experiment: The final experiment with this group of number of abstracts is fine grain model for "TRANSPORTATION" domain. Time taken to train this kind of model was in total 134.6 seconds, from which 126.98 seconds, spent in optimization. Testing routine is the same here.

Results of the experiment: From Table 3.71 we again have a lower result than the previous experiment with coarse grain model. But referencing to main experiment in Table 3.15, results now are closer to maximum value. As well the same situation is with experiments provided with global model in

Table 3.62 and Table 3.65 where those results there are lower than domain specific model results.

Entity	Precision	Recall	F1 score
Aircraft	1,0000	1,0000	1,0000
Automobile	1,0000	1,0000	1,0000
Infrastructure	1,0000	1,0000	1,0000
PublicTransitSystem	1,0000	0,9808	0,9903
Ship	1,0000	1,0000	1,0000
SpaceShuttle	1,0000	0,6667	0,8000
SpaceStation	1,0000	1,0000	1,0000
Train	1,0000	1,0000	1,0000
Totals	**1,0000**	**0,9735**	**0,9865**

Table 3.71: Results of ”TRANSPORTATION” domain specific model in fine grained run with 100 abstracts from the same domain

In conclusion we can say that models that were trained with lower number of abstracts than in main experiment, mostly provides a better results, of course the time needed to train those models and make experiments was faster, but here we don't have that much data unlike in main experiment, which if we try some other dataset who was not used to train the models, the results can we worst here than in models who has more trained data. Except the experiment where we had 100 abstracts to every domain, in most of other experiment, the fine grain model gives more precise recognition than the coarse grain model, which for us was quite surprise. And in the every group of experiment the domain specific models gives same or better results than global models and if we put there the time needed for training or testing domain specific models, then the results for now is that domain specific models are better for usage.

3.3.3 Experiments that have more than 300 abstracts in model and test files

In this subsection we have a two groups of number of abstracts retrieved from DBpedia, that are bigger than in the main experiment. One of them is where we have 400 abstracts per domain and the other one is with 500 abstracts per domain, which is the maximum that we succeeded to train.

3.3.3.1 Evaluation of global domain with 400 abstracts from every domain

Datasets with 400 abstracts for every domain. As well because we wanted to know the impact of train data who has more abstracts than the main experiment, we've increased the number of retrieved data to 400 abstracts per

domain. Our algorithm needs 184.92 minutes to retrieve data from DBpedia and prepare datasets ready to use in Stanford NER application.

Description of the experiment: In Table 3.72 we provide an experiment where the model was trained with abstracts from every domain, in total 1200 abstracts, annotated in coarse grain. We need 1041.3 seconds to train the model, from which 1008.25 seconds spent in optimization. The model was tested with the same dataset that was trained.

Results of the experiment: Table 3.72 show the output of the experiment, where the results to every domain are close to the maximum values on every measurement. We can say that for such big model the results are fantastic. Referring to the main experiment from Table 3.2, now this kind of model provide slightly lower results, but we have more trained data here.

Entity	**Precision**	**Recall**	**F1 score**
POLITICS	0,9804	0,9434	0,9615
SPORT	0,9832	0,9590	0,9709
TRANSPORTATION	0,9941	0,9754	0,9847
Totals	**0,9849**	**0,9584**	**0,9714**

Table 3.72: Results of global model in coarse grained run with 400 abstracts from every domain

Description of the experiment: For purposes of the experiment in Table 3.73 we have use the same global trained model from the previous experiment, but now the test file contains only 400 abstracts from "POLITICS" domain.

Results of the experiment: From Table 3.73 we see that the result is way more worst than the previous experiment, also model recognizes entities from other domain which are not part of dataset.If we compare only result of "POLITICS" type with the previous experiment we can see that now results are worst then there. Also in comparing with the main experiment from Table 3.3, now model gives a very very little better results, although recognize entities from "SPORT" domain, which is not the case in main experiment.

Entity	**Precision**	**Recall**	**F1 score**
POLITICS	0,9754	0,4082	0,5756
SPORT	0,0000	1,0000	0,0000
TRANSPORTATION	0,0000	1,0000	0,0000
Totals	**0,9531**	**0,4082**	**0,5716**

Table 3.73: Results of global model in coarse grained run with 400 abstracts from "POLITICS" domain

Description of the experiment: This experiment is almost identical

like previous one, with only difference is test file, where now we tested with abstracts from ”SPORT” domain.

Results of the experiment: From Table 3.74 we see that model despite ”SPORT” type, recognize entities from other 2 domain which are not part of dataset, This brings results a little bit lower than the results of ”SPORT” type. We have the same situation on the main experiment in Table 3.4, but now results there are better for a bit. As well if we get only results for ”SPORT” type and compare with the results from global model in Table 3.72 we can see that now results are again a bit better.

Entity	**Precision**	**Recall**	**F1 score**
POLITICS	0,0000	1,0000	0,0000
SPORT	0,9837	0,9588	0,9711
TRANSPORTATION	0,0000	1,0000	0,0000
Totals	**0,9805**	**0,9588**	**0,9695**

Table 3.74: Results of global model in coarse grained run with 400 abstracts from ”SPORT” domain

Description of the experiment: The final experiment is also the same like previous two, where now test file contains only abstracts from ”TRANSPORTATION” domain.

Results of the experiment: Table 3.75 shows that for ”TRANSPORTATION” entities we have precision closer to maximum value. But how in previous 2 experiment, also here model recognize entities which are not part of test dataset. Now if we compare only the results of ”TRANSPORTATION” type with the results in global model from Table 3.72 we see that global model provides a slight better results. Referring to the main experiment from Table 3.5, the overall results of the experiments are the same, but now model recognize entities also with ”POLITICS” type and the result for ”TRANSPORTATION” type are as well a bit lower.

Entity	**Precision**	**Recall**	**F1 score**
POLITICS	0,0000	1,0000	0,0000
SPORT	0,0000	1,0000	0,0000
TRANSPORTATION	0,9939	0,9762	0,9806
Totals	**0,9861**	**0,9822**	**0,9841**

Table 3.75: Results of global model in coarse grained run with 400 abstracts from ”TRANSPORTATION” domain

Description of the experiment: Experiment in Table 3.76 is provided with same data like the experiment in Table 3.72, but now the model and test data are annotated in fine grained. We needed in total 5139.7 seconds to train model, from which 5095.94 seconds spent in optimization.

Results of the experiment: How we can see from Table 3.76 our model provide maximum precision on most of entity types and maximum recall on some entity types. This helps to have a bit better results the experiment in coarse grain model from Table 3.72. Also in comparing with the main experiment from Table 3.6 we have a slightly lower results here, but a more annotated entities.

Entity	**Precision**	**Recall**	**F1 score**
Aircraft	1,0000	1,0000	1,0000
Athlete	1,0000	0,9899	0,9949
Automobile	1,0000	1,0000	1,0000
Coach	1,0000	1,0000	1,0000
Infrastructure	1,0000	0,9896	0,9948
PoliticalParty	0,9766	0,9486	0,9624
Politician	1,0000	0,9893	0,9946
PublicTransitSystem	0,9935	0,9776	0,9855
Ship	1,0000	0,9231	0,9600
SpaceShuttle	1,0000	1,0000	1,0000
SpaceStation	1,0000	1,0000	1,0000
SportsClub	0,9796	0,9658	0,9726
SportsEvent	1,0000	0,8636	0,9268
SportsLeague	0,9698	0,9835	0,9766
SportsManager	1,0000	0,9726	0,9861
SportsTeam	1,0000	0,9851	0,9925
Train	1,0000	1,0000	1,0000
Totals	**0,9870**	**0,9709**	**0,9789**

Table 3.76: Results of global model in fine grained run with 400 abstracts from every domain

Description of the experiment: In this experiment we use the same trained model from previous experiment, but now the test file contains only abstracts from ”POLITICS” domain, who is annotated in fine grain.

Results of the experiment: Table 3.77 show the output of the experiment, where we can see that even we have Election type on model and test file, the model do not find any entity with that type. Also for the Politician type we have a very low recall, which reflects that there is a very low overall result. Here we can also see which types the model recognize from other 2 domains, who are not part of the dataset. This also contributes to lower result.

In comparing with the experiment in coarse grain (see Table 3.73, now we have a bit better overall result. As well the results here are better than in main experiment from Table 3.7, even the model now recognize more types than there.

Entity	Precision	Recall	F1 score
Aircraft	0,0000	1,0000	0,0000
Election	0,0000	0,0000	0,0000
PoliticalParty	0,9766	0,9484	0,9623
Politician	1,0000	0,2092	0,3460
PublicTransitSystem	0,0000	1,0000	0,0000
Ship	0,0000	1,0000	0,0000
SportsClub	0,0000	1,0000	0,0000
SportsLeague	0,0000	1,0000	0,0000
Totals	**0,9619**	**0,4151**	**0,5799**

Table 3.77: Results of global model in fine grained run with 400 abstracts from ”POLITICS” domain

Description of the experiment: How in the previous experiment also here we have the same model but now tested with abstracts from ”SPORT” domain annotated in fine grain.

Results of the experiment: In Table 3.78 we have the output of the provided experiment. How we can see the results are not bad at all for such big data. Comparable with the coarse grain model in Table 3.74 now we have a better overall result, although model also recognize wrong entities. Referencing to main experiment in Table 3.8 the results there are bit lower than now, even though than model recognize more wrong entities than in main experiment.

Entity	Precision	Recall	F1 score
Aircraft	0,0000	1,0000	0,0000
Athlete	1,0000	0,9899	0,9949
Coach	1,0000	1,0000	1,0000
PoliticalParty	0,0000	1,0000	0,0000
Politician	0,0000	1,0000	0,0000
SportsClub	0,9794	0,9654	0,9724
SportsEvent	1,0000	0,8636	0,9268
SportsLeague	0,9696	0,9834	0,9765
SportsManager	1,0000	0,9726	0,9861
SportsTeam	1,0000	0,9850	0,9924
Train	0,0000	1,0000	0,0000
Totals	**0,9821**	**0,9721**	**0,9770**

Table 3.78: Results of global model in fine grained run with 400 abstracts from ”SPORT” domain

Description of the experiment: The last experiment with the model used in previous 3 experiment is now the dataset test file with abstracts from

"TRANSPORTATION" domain also annotated in fine grain.

Results of the experiment: Table 3.79 shows that our train model provide a bit better results than the coarse grain experiment in Table 3.75. Also comparing with the main experiment from Table 3.9, results there are little bit better than now. Than can be because model recognize one more wrong entity (PoliticalParty entity).

Entity	Precision	Recall	F1 score
Aircraft	1,0000	1,0000	1,0000
Automobile	1,0000	1,0000	1,0000
Infrastructure	1,0000	0,9896	0,9948
PoliticalParty	0,0000	1,0000	0,0000
Politician	0,0000	1,0000	0,0000
PublicTransitSystem	0,9934	0,9773	0,9853
Ship	1,0000	1,0000	1,0000
SpaceShuttle	1,0000	1,0000	1,0000
SpaceStation	1,0000	1,0000	1,0000
SportsClub	0,0000	1,0000	0,0000
SportsTeam	0,0000	1,0000	0,0000
Train	1,0000	1,0000	1,0000
Totals	**0,9866**	**0,9866**	**0,9866**

Table 3.79: Results of global model in fine grained run with 400 abstracts from "TRANSPORTATION" domain

3.3.3.2 Evaluation of domain specific models with 400 abstracts

Description of the experiment: Experiment in Table 3.80 was provided with model trained only with abstracts from "POLITICS" domain in coarse grain. To train this model we needed 125.6 seconds, from which 114.88 seconds spent in optimization. The model is tested with the same dataset that is created.

Results of the experiment: How we can see from Table 3.80 the result is not bad at all for this big model. In comparing with experiments provided with global domain in Table 3.72 and in Table 3.73, the result now is better and a significant difference in result we can see in experiment where we had a global model tested with same dataset like here (see Table 3.73). Also referencing to main experiment from Table 3.10 where the only difference is the number of abstracts used for training the model, now the result is a better than there, although that now we have more data.

Entity	Precision	Recall	F1 score
POLITICS	0,9866	0,9479	0,9669
Totals	**0,9866**	**0,9479**	**0,9669**

Table 3.80: Results of ”POLITICS” domain specific model in coarse grained run with 400 abstracts from the same domain

Description of the experiment: For the purposes of this experiment we have used the same data from the previous one, but now annotated in fine grain. To train this kind of model we needed 416.7 seconds, from which 405.16 seconds spent in optimization.

Results of the experiment: We tested the model with the same data that was created and how we can see from Table 3.81 the overall result is better than previous experiment with coarse grain model. Also in comparing with the main experiment from Table 3.11 now model provides also a better result, but not that significant like in experiment from Table 3.77.

Entity	Precision	Recall	F1 score
Election	0,9975	0,9590	0,9779
PoliticalParty	0,9767	0,9530	0,9647
Politician	0,9977	0,9920	0,9948
Totals	**0,9906**	**0,9717**	**0,9810**

Table 3.81: Results of ”POLITICS” domain specific model in fine grained run with 400 abstracts from the same domain

Description of the experiment: Experiment in Table 3.82 is provided with a coarse grain model trained with abstracts only from ”SPORT” domain. The time needed to train this model was 90.9 seconds, from which 81.58 seconds spent in optimization. To test it we have used the same dataset that model was trained.

Results of the experiment: How we can see from Table 3.82 the trained model give a worthy results, who in comparing with the experiments provided with global model in Table 3.72 and Table 3.74 are a bit better. As well comparing with the results from the main experiment (see Table 3.12) we see that now results are also quite better, although we have a more entities.

Entity	Precision	Recall	F1 score
SPORT	0,9858	0,9676	0,9766
Totals	**0,9858**	**0,8676**	**0,9766**

Table 3.82: Results of ”SPORT” domain specific model in coarse grained run with 400 abstracts from the same domain

Description of the experiment: In this experiment we have used the same dataset from previous, but now entities are annotated in fine grain. To train a fine grain model we have need 915.1 seconds, from which 898.50 second spent in optimization. As well as previous the model is tested with the dataset that is trained.

Results of the experiment: In Table 3.83 we see the output of the experiment. Model give maximum precision on most of entities, as well the recall values are not bad at all but only one entity has maximum result. This results with a bit lower F1 score than the maximum value. In comparing with the previous experiment now overall result is a little bit better. Also comparing with the experiment with a global model in Table 3.78, again results are better. Final comparison is with main experiment in Table 3.13, where as well the result is significantly better now.

Entity	**Precision**	**Recall**	**F1 score**
Athlete	1,0000	0,9731	0,9863
Coach	1,0000	1,0000	1,0000
SportsClub	0,9815	0,9715	0,9765
SportsEvent	1,0000	0,9091	0,9524
SportsLeague	0,9718	0,9787	0,9752
SportsManager	1,0000	0,9726	0,9850
SportsTeam	1,0000	0,9900	0,9796
Totals	**0,9865**	**0,9727**	**0,9796**

Table 3.83: Results of "SPORT" domain specific model in fine grained run with 400 abstracts from the same domain

Description of the experiment: At the end we train a model with abstracts from "TRANSPORTATION" domain. To train a coarse grain model from this domain we needed 72.5 seconds, from which 64.53 seconds spent in optimization. Of course, here also we tested the model with the same dataset that was created.

Results of the experiment: In Table 3.84 we see that the model, unlike in previous experiment, gives values from every measurement close to maximum value. When we compare the results with the results from experiments provided with the global model in Table 3.72 and Table 3.75 we can see that now we again have a bit better result than in those experiments. Referring to main experiment in Table 3.14, the results are as well significantly better and more usable.

Entity	Precision	Recall	F1 score
TRANSPORTATION	0,9954	0,9747	0,9850
Totals	**0,9954**	**0,9747**	**0,9850**

Table 3.84: Results of "TRANSPORTATION" domain specific model in coarse grained run with 400 abstracts from the same domain

Description of the experiment: We also train a fine grain model from "TRANSPORTATION" domain. To train it we needed 725.2 seconds, from which 711.78 seconds spent in optimization. As well as the previous experiment, the model is tested with the same dataset that is created.

Results of the experiment: From the output of the experiment in Table 3.85 we can see that model provides excellent results, despite that, that is tested with big dataset. Comparing with the experiment who was provided with the global model and tested with same dataset like here in Table 3.79, now we have a bit better results, but a significant improvement on the results we have when we refer to the main experiment from Table 3.15.

Entity	Precision	Recall	F1 score
Aircraft	1,0000	0,9835	0,9917
Automobile	1,0000	0,9583	0,9787
Infrastructure	1,0000	0,9948	0,9974
PublicTransitSystem	0,9934	0,9773	0,9853
Ship	1,0000	1,0000	1,0000
SpaceShuttle	1,0000	0,6667	0,8000
SpaceStation	1,0000	1,0000	1,0000
Train	1,0000	1,0000	1,0000
Totals	**0,9970**	**0,9807**	**0,9888**

Table 3.85: Results of "TRANSPORTATION" domain specific model in fine grained run with 400 abstracts from the same domain

3.3.3.3 Evaluation of global domain with 500 abstracts from every domain

Datasets with 500 abstracts for every domain. In this final group of experiments with 500 abstracts per domain we will repeat all experiments unlike in previous groups. To retrieve and prepare datasets for training in Stanford NER, our algorithm needs in total 3.63 hours.

Description of the experiment: In Table 3.86 we provide an experiment where the model is trained with, in total, 1500 abstracts annotated in coarse grain. To train this kind of model we needed 1021.3 seconds, from which 989.89 seconds spent in optimization. The model is tested with the dataset that was trained.

Results of the experiment: As we can see from Table 3.86 model provides amazing results even though that we have now a lot of data. From here we can see that results are closer to the main experiment from Table 3.2, but results there are bit better than now.

Entity	Precision	Recall	F1 score
POLITICS	0,9788	0,9444	0,9613
SPORT	0,9850	0,9596	0,9721
TRANSPORTATION	0,9962	0,9750	0,9855
Totals	**0,9857**	**0,9587**	**0,9720**

Table 3.86: Results of global model in coarse grained run with 500 abstracts from every domain

Description of the experiment: For purposes of this experiment we have used the model train previously, but now it is tested with the dataset that contains only abstracts from "POLITICS" domain, annotated in coarse grain.

Results of the experiment: From Table 3.87 we see that even the precision is not that bad, but because of very low recall, the F1 score is lower, which means that model recognize only half of our entities. Also model recognize entities that are not part of the tested dataset. As well when we compare with the previous experiment is clearly that there the results for "POLITICS" domain are way more better. Also comparably with the main experiment from Table 3.3 results now are bit better, although there model recognize only some entities from "TRANSPORTATION" domain.

Entity	Precision	Recall	F1 score
POLITICS	0,9734	0,4095	0,5765
SPORT	0,0000	1,0000	0,0000
TRANSPORTATION	0,0000	1,0000	0,0000
Totals	**0,9549**	**0,4095**	**0,5732**

Table 3.87: Results of global model in coarse grained run with 500 abstracts from "POLITICS" domain

Description of the experiment: In this experiment we also used the global trained model, but now it is tested with "SPORT" domain abstracts dataset.

Results of the experiment: Table 3.88 shows the output of the experiment, from where we see that even the results aren't at their maximum are quite satisfying for such a big model and dataset. Also here model recognize entities that are not part of the dataset. As well in comparing with the results for "SPORT" type in global model, now we have a very very little lower

result. The same situation is with the main experiment in Table 3.4, where again results there are bit better.tter now.

Entity	Precision	Recall	F1 score
POLITICS	0,0000	1,0000	0,0000
SPORT	0,9849	0,9594	0,9720
TRANSPORTATION	0,0000	1,0000	0,0000
Totals	**0,9788**	**0,9594**	**0,9690**

Table 3.88: Results of global model in coarse grained run with 500 abstracts from "SPORT" domain

Description of the experiment: Final experiment with the global domain is that is tested with the dataset that contains only abstracts from "TRANSPORTATION" domain.

Results of the experiment: From Table 3.89 we see that model again recognize some entities from other 2 domains, that are not part of the tested dataset. Also if we compare only results for "TRANSPORTATION" domain with the results from global experiment in Table 3.86, they are almost the same without any significant difference. The situation in main experiment from Table 3.5 is a little bit different, because there model recognize only one wrong entity and the results are bit better.

Entity	Precision	Recall	F1 score
POLITICS	0,0000	1,0000	0,0000
SPORT	0,0000	1,0000	0,0000
TRANSPORTATION	0,9961	0,9769	0,9864
Totals	**0,9870**	**0,9769**	**0,9819**

Table 3.89: Results of global model in coarse grained run with 500 abstracts from "TRANSPORTATION" domain

Description of the experiment: For purposes of this and the next 3 experiments we train a fine grain global model with 500 abstracts from every domain. To train this kind of model we needed 6706.7 seconds, from which 6650.30 seconds spent in optimization. In this experiment we tested the model with the dataset that was trained.

Results of the experiment: From Table 3.90 we can see the results for every particular ontology type. So even we have a lot of entities we see that results are not bad at all, even model provides maximum precision on most of entities and maximum recall on some of entities, which for a big model is excellent. When we compare with the results from coarse grain model, now we have a bit better overall result. As well referencing to main experiment in Table 3.6 the results there are bit worst than here, although here we have more data.

Entity	Precision	Recall	F1 score
Aircraft	1,0000	0,9929	0,9964
Athlete	1,0000	0,9896	0,9948
Automobile	1,0000	1,0000	1,0000
Coach	1,0000	1,0000	1,0000
Infrastructure	1,0000	0,9783	0,9890
PoliticalParty	0,9775	0,9403	0,9585
Politician	1,0000	0,9874	0,9937
PublicTransitSystem	0,9944	0,9807	0,9875
Ship	1,0000	0,9259	0,9615
SpaceShuttle	1,0000	1,0000	1,0000
SpaceStation	1,0000	1,0000	1,0000
SportsClub	0,9756	0,9553	0,9654
SportsEvent	1,0000	0,8796	0,9360
SportsLeague	0,9700	0,9810	0,9755
SportsManager	1,0000	0,9780	0,9889
SportsTeam	1,0000	0,9831	0,9915
Train	1,0000	1,0000	1,0000
Totals	**0,9866**	**0,9669**	**0,9766**

Table 3.90: Results of global model in fine grained run with 500 abstracts from every domain

Description of the experiment: In this experiment we tested the global model with dataset that contains only abstracts from ”POLITICS” domain, of course annotated in fine grain.

Results of the experiment: Table 3.91 shows the output of the experiment, where model provides terrible results and also recognize entity types which is not part of the dataset, but don’t recognize any entity with Election type who is part of the dataset. This model for consolation gives a bit better results than the experiment with coarse grain model in Table 3.87. As well referring to main experiment in Table 3.7, now model gives a bit better results, but when we take also the size of the model this is a huge difference.

Entity	Precision	Recall	F1 score
Aircraft	0,0000	1,0000	0,0000
Election	0,0000	0,0000	0,0000
PoliticalParty	0,9774	0,9400	0,9583
Politician	1,0000	0,2171	0,3567
PublicTransitSystem	0,0000	1,0000	0,0000
Ship	0,0000	1,0000	0,0000
SportsClub	0,0000	1,0000	0,0000
SportsLeague	0,0000	1,0000	0,0000
Totals	**0,9631**	**0,4138**	**0,5789**

Table 3.91: Results of global model in fine grained run with 500 abstracts from ”POLITICS” domain

Description of the experiment: Here we also have the global train model, but for this experiment is tested with dataset abstracts from ”SPORT” domain.

Results of the experiment: As we see from Table 3.92 model gives slightly better results than in the experiment with coarse grain model from Table 3.86. As well referencing to main experiment in Table 3.8, now model provides a bit lower result.

Entity	Precision	Recall	F1 score
Aircraft	0,0000	1,0000	0,0000
Athlete	1,0000	0,9896	0,9948
Coach	1,0000	1,0000	1,0000
PoliticalParty	0,0000	1,0000	0,0000
Politician	0,0000	1,0000	0,0000
SportsClub	0,9753	0,9549	0,9650
SportsEvent	1,0000	0,8796	0,9360
SportsLeague	0,9717	0,9810	0,9763
SportsManager	1,0000	0,9780	0,9889
SportsTeam	1,0000	0,9831	0,9915
Train	0,0000	1,0000	0,0000
Totals	**0,9785**	**0,9690**	**0,9737**

Table 3.92: Results of global model in fine grained run with 500 abstracts from ”SPORT” domain

Description of the experiment: The final experiment with the global model is with dataset that contains only abstracts from ”TRANSPORTATION” domain.

Results of the experiment: Table 3.93 shows the outcome of the experiment, where for the types from ”TRANSPORTATION” domain we have an

excellent results, but because model also recognize wrong entities, the overall results is lower. But in comparing with the experiment with coarse grain model in Table 3.89, now we have a bit better results. As well referring to main experiment in Table 3.9, we now have again a bit lower result.

Entity	Precision	Recall	F1 score
Aircraft	1,0000	0,9927	0,9963
Automobile	1,0000	1,0000	1,0000
Infrastructure	1,0000	0,9783	0,9890
PoliticalParty	0,0000	1,0000	0,0000
Politician	0,0000	1,0000	0,0000
PublicTransitSystem	0,9943	0,9804	0,9873
Ship	1,0000	1,0000	1,0000
SpaceShuttle	1,0000	1,0000	1,0000
SpaceStation	1,0000	1,0000	1,0000
SportsClub	0,0000	1,0000	0,0000
SportsTeam	0,0000	1,0000	0,0000
Train	1,0000	1,0000	1,0000
Totals	**0,9884**	**0,9833**	**0,9858**

Table 3.93: Results of global model in fine grained run with 500 abstracts from "TRANSPORTATION" domain

3.3.3.4 Evaluation of domain specific models with 500 abstracts

Description of the experiment: In this experiment we train a domain specific coarse grain model with dataset that contains only abstracts from "POLITICS" domain. The time needed to train this model was 165.6 seconds, from which 152.45 seconds spent in optimization. The model is tested with the same dataset that is trained.

Results of the experiment: As we can see from Table 3.94 this model provides a better result than the 2 experiments with the global model in Table 3.86 and and a way more better results than experiment from Table 3.87. As well referencing to main experiment in Table 3.10, now model gives better results.

Entity	Precision	Recall	F1 score
POLITICS	0,9808	0,9450	0,9626
Totals	**0,9808**	**0,9450**	**0,9626**

Table 3.94: Results of "POLITICS" domain specific model in coarse grained run with 500 abstracts from the same domain

textbfDescription of the experiment: With the same data from the previous experiment, but not annotated in fine grain, we train new model. We

needed 479.3 seconds to train this model, from which 465.16 seconds spent in optimization. The model is tested with the same dataset that was trained.

Results of the experiment: Table 3.95 shows the outcome of the experiment, where we see that even results on every type are close to maximum, the overall results is a bit better than the previous experiment. But in comparing to the experiment with global domain inTable 3.91, now results are way more better, we don't have any wrong recognition and also entities from Election ontology type are recognized. Referencing to the main experiment in Table 3.11, as well now model provides a better results, which is surprisingly for such big dataset.

Entity	**Precision**	**Recall**	**F1 score**
Election	0,9915	0,9393	0,9647
PoliticalParty	0,9777	0,9502	0,9637
Politician	0,9962	0,9877	0,9919
Totals	**0,9890**	**0,9648**	**0,9768**

Table 3.95: Results of "POLITICS" domain specific model in fine grained run with 500 abstracts from the same domain

Description of the experiment: For the purposes of the experiment we train a coarse grain model with dataset that contains only abstracts from "SPORT" domain. Time needed to train this kind of model was 115.0 seconds, from which 103.59 seconds spent in optimization. The model is tested like in previous experiments, with the same dataset that is trained.

Results of the experiment: From Table 3.96 we see that model gives a slightly better results than in experiments with global domain in Table 3.86 and Table 3.88. Also referencing to the main experiment in Table 3.12, now model gives again significant better results than there.

Entity	**Precision**	**Recall**	**F1 score**
SPORT	0,9856	0,9706	0,9780
Totals	**0,9856**	**0,9706**	**0,9780**

Table 3.96: Results of "SPORT" domain specific model in coarse grained run with 500 abstracts from the same domain

Description of the experiment: This experiment is provided with a "SPORT" fine grain model. To train this model we needed 1175.1 seconds, from which 1158.90 seconds spent in optimization. Test is the same like previous experiment, where dataset now are annotated in fine grain.

Results of the experiment: As we see from Table 3.92 the overall results is a bit better than previous experiment, as well better than experiment provided with the global model in Table 3.92. Also referencing to main ex-

periment in Table 3.13, now model was more precise than there, an because of that gives better results.

Entity	Precision	Recall	F1 score
Athlete	1,0000	0,9791	0,9894
Coach	1,0000	1,0000	1,0000
SportsClub	0,9771	0,9630	0,9700
SportsEvent	1,0000	0,9074	0,9515
SportsLeague	0,9755	0,9848	0,9801
SportsManager	1,0000	0,9780	0,9889
SportsTeam	1,0000	0,9915	0,9957
Totals	**0,9861**	**0,9731**	**0,9796**

Table 3.97: Results of "SPORT" domain specific model in fine grained run with 500 abstracts from the same domain

Description of the experiment: Experiment in Table 3.98 is provided with a coarse grain model train with dataset from "TRANSPORTATION" domain. To train this model we needed 78.3 seconds in total, from which 67.96 seconds spent in optimization. As previous model is tested with same dataset that is trained.

Results of the experiment: Table 3.98 shows that model provides very good results on every measurement. Comparably with experiments with global model in Table 3.86 and Table 3.89, now model gives better overall score than there and it's faster to train and provide experiment. As well referencing to main experiment in Table 3.14 again results from this experiment are significantly better than main experiment results.

Entity	Precision	Recall	F1 score
TRANSPORTATION	0,9974	0,9756	0,9864
Totals	**0,9974**	**0,9756**	**0,9864**

Table 3.98: Results of "TRANSPORTATION" domain specific model in coarse grained run with 500 abstracts from the same domain

Description of the experiment: The final experiment with this group of number of abstracts is fine grain model for "TRANSPORTATION" domain. Time taken to train this kind of model was in total 1040.2 seconds, from which 1023.55 seconds, spent in optimization. Testing routine is the same here.

Results of the experiment: From Table 3.99 we again have excellent results. Comparing with the experiment provided with global model in Table 3.93 and the main experiment in Table 3.15, model here gives a better results, than in those experiments.

Entity	Precision	Recall	F1 score
Aircraft	1,0000	0,9781	0,9889
Automobile	1,0000	0,8800	0,9362
Infrastructure	1,0000	0,9870	0,9934
PublicTransitSystem	0,9915	0,9804	0,9860
Ship	1,0000	1,0000	1,0000
SpaceShuttle	1,0000	0,6667	0,8000
SpaceStation	1,0000	1,0000	1,0000
Train	1,0000	1,0000	1,0000
Totals	**0,9961**	**0,9769**	**0,9864**

Table 3.99: Results of "TRANSPORTATION" domain specific model in coarse grained run with 500 abstracts from the same domain

In this group of experiment where trained models and datasets contains more abstracts than in the main experiment, we had a quite surprisingly results. In most of the experiments the results where better than in the main experiment, which when we compare the size of the data that makes a huge difference on recognized entities. As well we had some results where the results where a bit lower than in the main experiment, but again here comes the size of data, which as we say, we have a way more recognized entities.

Our assumptions of that, that a domain specific models will provide a better results were true. In all of the experiment the domain specific model gives a bit better results, as well those domain takes less time to train and test. Another thing was that the fine grain model, even takes a bit more time to train, also in this group gives a better results than coarse grain models. This can be handy, because on fine grain models we see all entities and how they perform.

3.3.4 Evaluation of domains tested with two or more datasets

Description of experiment: This experiment is provided with the global train model in fine grain from the main experiment. Model is tested with the two datasets. One dataset contains 500 abstracts per domain where fall also those 300 abstracts from the model, and the other dataset contains also 500 abstracts, but now those abstracts has a lower PageRank.

Results of the experiment: As we can see from Table 3.100 for some entities we have a maximum precision, conversely for some entities model do not find nothing, because those entities maybe are from the dataset which model do not contain them. As well the recall on founded entities is very low, and the reason is same like in precision measurement.

Entity	Precision	Recall	F1 score
Aircraft	0,9242	0,5755	0,7093
Athlete	0,8182	0,3778	0,5169
Automobile	0,9565	0,3607	0,5238
Coach	1,0000	0,2000	0,3333
Infrastructure	1,0000	0,9896	0,9948
Locomotive	0,0000	0,0000	0,0000
Motorcycle	0,0000	0,0000	0,0000
OrganisationMember	0,0000	0,0000	0,0000
PoliticalParty	0,7656	0,5819	0,6613
Politician	0,8925	0,4099	0,5618
PublicTransitSystem	0,8291	0,6178	0,7080
Ship	0,9375	0,3409	0,5000
SpaceShuttle	1,0000	0,3750	0,5455
SpaceStation	1,0000	0,3333	0,5000
SportsClub	0,8009	0,4276	0,5575
SportsEvent	0,9559	0,3171	0,4762
SportsLeague	0,8071	0,5912	0,6824
SportsManager	0,9643	0,2903	0,4463
SportsTeam	0,8856	0,5838	0,7037
Train	1,0000	0,5455	0,7059
Totals	**0,8206**	**0,4837**	**0,6087**

Table 3.100: Results of fine grain global model trained 300 abstracts per domain, tested with dataset that contains 500 abstracts, but with lower PageRank on article and dataset that contains 500 abstracts, but with higher PageRank.

Description of experiment: This experiment is provided with the global train model in fine grain that was trained with 500 abstracts per domain. Model is tested with the two datasets. One dataset contains 500 abstracts per domain, so the same dataset that model is trained, and the other dataset contains also 500 abstracts, but now those abstracts has a lower PageRank.

Results of the experiment: As we can see from Table 3.102 the results now are bit better than the previous experiment, but still we are bit higher than middle values, but not that close to maximum like in experiments where the model were tested with one dataset. This is caused by the fact that one of the dataset was not part of training the model, although the entity types are same. Another fact is the size of the models, so because of that model makes wrong recognition.

Entity	Precision	Recall	F1 score
Aircraft	0,9735	0,6934	0,8099
Athlete	0,9101	0,6222	0,7391
Automobile	1,0000	0,4098	0,5814
Coach	1,0000	0,3000	0,4615
Infrastructure	0,8885	0,5218	0,6575
Locomotive	0,0000	0,0000	0,0000
Motorcycle	0,0000	0,0000	0,0000
OrganisationMember	0,0000	0,0000	0,0000
PoliticalParty	0,8393	0,7403	0,7876
Politician	0,9271	0,6593	0,7706
PublicTransitSystem	0,9027	0,7389	0,8126
Rocket	0,0000	0,0000	0,0000
Ship	0,9615	0,5682	0,7143
SpaceShuttle	1,0000	0,4375	0,6087
SpaceStation	1,0000	0,6667	0,8000
SportsClub	0,8722	0,6071	0,7159
SportsEvent	0,9000	0,4829	0,6286
SportsLeague	0,8622	0,7357	0,7939
SportsManager	0,9787	0,4946	0,6571
SportsTeam	0,9276	0,7514	0,8302
Train	1,0000	0,5455	0,7059
Totals	**0,8844**	**0,6592**	**0,7553**

Table 3.101: Results of fine grain global model trained 500 abstracts per domain, tested with dataset that contains 500 abstracts, but with lower PageRank on article and dataset that contains 500 abstracts, but with higher PageRank.

Description of experiment: For purposes of this experiment we have used the same trained model from the previous one, but now it is tested with the dataset that contains 500 abstracts per domain, but with lower PageRank.

Results of the experiment: As we can see from Table 3.102 the results are not brilliant at all. Here we see the difference where model is tested with the completely different data that is trained. We see that maximum F1 score is 0.5154 for PublicTransitSystem entitites.

Entity	Precision	Recall	F1 score
Aircraft	0,6667	0,1111	0,1905
Athlete	0,4675	0,1268	0,1994
Automobile	0,0000	0,0000	0,0000
Coach	0,0000	0,0000	0,0000
Infrastructure	0,5352	0,1387	0,2203
Locomotive	0,0000	0,0000	0,0000
Motorcycle	0,0000	0,0000	0,0000
OrganisationMember	0,0000	0,0000	0,0000
PoliticalParty	0,5462	0,4097	0,4682
Politician	0,5962	0,1867	0,2844
PublicTransitSystem	0,6943	0,4098	0,5154
Rocket	0,0000	0,0000	0,0000
Ship	0,0000	0,0000	0,0000
SpaceShuttle	0,0000	0,0000	0,0000
SpaceStation	0,0000	0,0000	0,0000
SportsClub	0,6370	0,2675	0,3768
SportsEvent	0,2667	0,0412	0,0714
SportsLeague	0,6395	0,4125	0,5015
SportsManager	0,6000	0,0316	0,0600
SportsTeam	0,6316	0,2975	0,4045
Train	0,0000	0,0000	0,0000
Totals	**0,5983**	**0,2670**	**0,3692**

Table 3.102: Results of fine grain global model trained 500 abstracts per domain, tested with dataset that contains 500 abstracts, but with lower PageRank on article.

Description of experiment: In this experiment we have used a fine grain model trained with 500 abstracts only from "TRANSPORTATION" domain. The model now is tested with dataset that has 900 abstracts, that means 300 abstracts per domain and the dataset that has 300 abstracts only from "TRANSPORTATION" domain.

Results of the experiment: As we can see from Table 3.103 for the entities of "TRANSPORTATION" domain we have nice results, but because we tested with the dataset that has all abstracts the overall results is around middle value. As well model do not recognize any wrong entities from other domains, which is also excellent.

Entity	Precision	Recall	F1 score
Aircraft	0,9950	0,9706	0,9826
Athlete	0,0000	0,0000	0,0000
Automobile	1,0000	0,8500	0,9189
Coach	0,0000	0,0000	0,0000
Infrastructure	1,0000	0,9820	0,9909
PoliticalParty	0,0000	0,0000	0,0000
Politician	0,0000	0,0000	0,0000
PublicTransitSystem	0,9835	0,9676	0,9755
Ship	1,0000	0,9655	0,9825
SpaceShuttle	1,0000	0,6667	0,8000
SpaceStation	1,0000	1,0000	1,0000
SportsClub	0,0000	0,0000	0,0000
SportsEvent	0,0000	0,0000	0,0000
SportsLeague	0,0000	0,0000	0,0000
SportsManager	0,0000	0,0000	0,0000
SportsTeam	0,0000	0,0000	0,0000
Train	1,0000	0,9091	0,9524
Totals	**0,9909**	**0,3491**	**0,5163**

Table 3.103: Result of ”TRANSPORTATION” fine grained Top 500 Links tested with global dataset that contains 300 abstracts per domain and ”TRANSPORTATION” fine grained dataset with 300 abstracts

As we can see from the previous 4 experiments, it really depends on that how we choose the datasets and also on the size of the model. Also those experiments shows that if model is trained with one data and is tested with completely different data, the results are of course very low. Maybe if the model was trained with more abstracts and then tested, the results will be better. But because we have not enough RAM memory we not succeed to train a bigger model.

3.3.5 Evaluation of model who are trained with 500 abstracts and are tested with texts from news papers

In this section we wanted to know who the trained models will behaves when they are tested with texts from daily life, or in this case texts from BCC and CNN web page. We make a datasets for every domain. Those datasets contains 3 texts per domain. As well we choose a fine grain, because from the previous experiments we noticed that those models gives better results.

BBC

For the purposes of this experiment we have used BBC articles [26] [27] [28] [29] [30] [31]

Description of the experiment: For purposes of this experiment we used a fine grain model who was trained with 500 abstracts per domain, which is our biggest trained model. We tested it with the dataset than contains texts from BBC website. This dataset has 2 texts for every domain.

Results of the experiment: As we can see from Table 3.104 the results are not satisfying at all. Even a such a big train model it is not able to recognize all entities. It's true that we don't have a lot annotated words in dataset, but we still expected higher results.

Entity	Precision	Recall	F1 score
Athlete	1,0000	0,0303	0,0588
Infrastructure	1,0000	0,1818	0,3077
PoliticalFunction	0,0000	0,0000	0,0000
PoliticalParty	1,0000	0,1538	0,2667
Politician	0,0000	0,0000	0,0000
PublicTransitSystem	1,0000	0,3333	0,5000
SportsEvent	0,0000	0,0000	0,0000
SportsLeague	0,0000	0,0000	0,0000
SportsTeam	0,0000	0,0000	0,0000
Totals	**1,0000**	**0,0357**	**0,0690**

Table 3.104: Results of fine grain model trained with 1500 abstracts, tested with text from BBC

Description of the experiment: In this experiment we get the model trained with 500 abstracts from "POLITICS" domain. We tested it with the text from the politics spare from BBC web site.

Results of the experiment: From Table 3.105 is clear that now we have a better results than in the previous experiment for "POLITICS" entities. Now model recognize Politician entities, which was not the case in the previous experiment. Because of this now results are better and we can see the power of domain specific models.

[26] https://www.bbc.com/sport/tennis/43964642
[27] https://www.bbc.com/sport/rugby-union/43969948
[28] https://www.bbc.com/news/election-us-2016-37855894
[29] https://www.bbc.com/news/world-us-canada-43453312
[30] https://www.bbc.co.uk/news/technology-43962881
[31] https://www.bbc.com/news/uk-wales-43913363

Entity	Precision	Recall	F1 score
Election	0,0000	0,0000	0,0000
PoliticalFunction	0,0000	0,0000	0,0000
PoliticalParty	1,0000	0,1538	0,2667
Politician	0,5000	0,0588	0,1053
Totals	**0,6250**	**0,0649**	**0,1176**

Table 3.105: Results of fine grain model trained with 1500 abstracts, tested with text from BBC based on sport domain

Description of the experiment: For this experiment we used the model trained with 500 abstracts from "SPORT" domain. As in previous experiment, also here, model is tested with texts from the same domain like it is trained.

Results of the experiment: Table 3.106 we see than we have exactly the same results like in the global model experiment (see Table 3.104), where only Athlete entities are recognized. So here the only improvement that we have is the time needed to train the model and we can be sure that here cannot be any wrong recognition from other domains.

Entity	Precision	Recall	F1 score
Athlete	1,0000	0,0303	0,0588
SportsClub	0,0000	0,0000	0,0000
SportsEvent	0,0000	0,0000	0,0000
SportsLeague	0,0000	0,0000	0,0000
SportsTeam	0,0000	1,0000	0,0000
Totals	**1,0000**	**0,0127**	**0,0250**

Table 3.106: Results of fine grain model trained with 1500 abstracts, tested with text from BBC

Description of the experiment: This experiment shows how the model trained with 500 abstracts from "TRANSPORTATION" domain will behave when it is tested with texts from same domain taken from BBC web site.

Results of the experiment: As we can see from Table 3.107 now model gives a worst results than the experiment in Table 3.104, where there model also recognize PublicTransportSystem entities, which is not the case now. In this experiment is clear that the global domain provides a better results, because recognize one more entity, which is a big step.

Entity	Precision	Recall	F1 score
PublicTransitSystem	0,0000	0,0000	0,0000
Infrastructure	1,0000	0,1818	0,3077
Totals	**1,0000**	**0,1429**	**0,2500**

Table 3.107: Results of fine grain model trained with 1500 abstracts, tested with text from BBC

CNN

For the purposes of this experiment we have used BBC articles [32] [33] [34] [35] [36] [37]

Description of the experiment: For purposes of this experiment we used the same model like in experiment in Table 3.104, but now the model is tested with texts from CNN web page.

Results of the experiment: As we can see from Table 3.108 the results are even worst that in experiment from Table 3.104. Here model recognize just Politician entity. So even a such big model cannot provide average results from texts from daily basis.

Entity	Precision	Recall	F1 score
Athlete	0,0000	0,0000	0,0000
GeopoliticalOrganization	0,0000	0,0000	0,0000
Infrastructure	0,0000	1,0000	0,0000
Politician	0,5000	0,0227	0,0435
SportsClub	0,0000	0,0000	0,0000
SportsEvent	0,0000	0,0000	0,0000
SportsLeague	0,0000	0,0000	0,0000
Totals	**0,1667**	**0,0139**	**0,0256**

Table 3.108: Results of fine grain model trained with 1500 abstracts, tested with text from CNN

Description of the experiment: In this experiment we also used the fine grain trained model with 500 abstracts from "POLITICS" domain. Model

[32] https://edition.cnn.com/2018/04/30/politics/trump-merkel-putin-advice/index.html

[33] https://m.cnn.com/en/article/h_53075a6fd222756ef530d84023dd8be7

[34] https://edition.cnn.com/2018/04/27/sport/tiger-woods-pga-tour-players-championship-spt/index.html

[35] https://edition.cnn.com/2018/04/24/sport/kentucky-derby-gronkowski-horse-racing-nfl-spt/index.html

[36] https://edition.cnn.com/travel/article/most-extreme-airports-world/index.html

[37] http://money.cnn.com/2018/05/02/news/companies/boeing-747-qantas-fleet/index.html

is tested with dataset than contains texts from the same domain from CNN web page.

Results of the experiment: As we see from Table 3.109 now model recognizes the same entity like the previous one, but now with a better score. So here as well we see the power of domain specific model.

Entity	**Precision**	**Recall**	**F1 score**
GeopoliticalOrganization	0,0000	0,0000	0,0000
Politician	0,8333	0,0893	0,1613
Totals	**0,8333**	**0,0877**	**0,1587**

Table 3.109: Results of fine grain model trained with 1500 abstracts, tested with text from CNN based on sport domain

Description of the experiment: For this experiment we used the model trained with 500 abstracts from "SPORT" domain. We tested the model with texts from the same domain from CNN web page.

Results of the experiment: In Table 3.110 we again see the power of domain specific model. We have a recognition on Athlete entities, which was not the case in experiment from Table 3.108. Even thought that the score is very low, we still have some improvements.

Entity	**Precision**	**Recall**	**F1 score**
Athlete	1,0000	0,0667	0,1250
SportsClub	0,0000	0,0000	0,0000
SportsEvent	0,0000	0,0000	0,0000
SportsLeague	0,0000	0,0000	0,0000
SportsTeam	0,0000	1,0000	0,0000
Totals	**0,5000**	**0,0370**	**0,0690**

Table 3.110: Results of fine grain model trained with 1500 abstracts, tested with text from CNN

Description of the experiment: For the purposes of this experiment we get the fine grain model trained with 500 abstracts from "TRANSPORTATION" domain. As in previous experiments, also here we test the model with the texts from same domain from CNN web page.

Results of the experiment: As we see from Table 3.111 for some reason model gives a maximum recall value to two entities. But, because there is no precision, model do not recognize any entity. This was also the case in experiment from Table 3.108, so here we don't have any improvements or looseness.

Entity	Precision	Recall	F1 score
Aircraft	0,0000	1,0000	0,0000
Infrastructure	0,0000	1,0000	0,0000
Totals	**0,0000**	**1,0000**	**0,0000**

Table 3.111: Results of fine grain model trained with 1500 abstracts, tested with text from BBC

From the provided 8 experiments, except the experiment in Table 3.107 where domain specific model gives worst results than a global model, in all other experiment we had a better or same results, which shows that the domain specific models are more usable. Another advantage is the time to train and test models.

3.3.6 Summary of the results

The global models from the main experiment which have 300 abstracts per domain, gives a slightly better results that the domain specific models, except the "POLITICS" domain specific models who have a way better results than a global model. As well the fine grain models provide a bit better results than a coarse grain models.

In the group of 10 abstracts per domain, the results from the global models and domain specific models were almost the same, except the "POLITICS" domain specific models which again give significantly better results than a global model. As well in this group of experiment the coarse and fine grain models give very similar results, without any significant difference.

The domain specific models from 20 abstracts per domain group, again gives a better results than the global models. But here the coarse grain models provide a bit better results than a fine grain models.

The same situation was for group of 40 abstracts per domain, where the fine grain domain specific models provide better results. But from this point, global models start to recognize wrong entities which were not part of the testing dataset.

In the group of 100 abstracts per domain, the models from a specific domain annotated in coarse grain provide better results than the global models in both annotation and better results than domain specific fine grain models.

The domain specific fine grain models from groups of 400 abstracts per domain and 500 abstracts per domain give slightly better results than the results from other types of experiments in particular group.

As well we test some models with the datasets that were not used for training the model or with bigger dataset than the model was trained. So the results from those experiments were worse than any previous experiments, which shows that results depends on data used to train and test model.

As summary from the provided experiments we can say that domain specific fine grain models give better results in comparing to domain specific coarse grain models and as well global models in both annotations. This give advantage because training a domain specific fine grain models takes less time, than global fine grain models. As well models trained with higher number of abstracts gives better results unlike models trained with lower number of abstracts.

Conclusion

The goal of this master thesis was to check does creating and testing domain specific models that will be used in NER application is a better solution, compared to global models who are used until now.

In the introduction of this thesis was explained which technologies was used to create this thesis and also common NER applications used today. In addition, the research related to this topic was included.

The whole process of preparing the datasets ready to be used in Stanford NER application was explained. The process of transforming downloaded raw data to data that are ready for processing to be able, in reasonable time, to create a datasets was also explained. The algorithm that was used for preparing datasets is also included. Secondly, we explained the way of choosing the domains that were used in this thesis. After that we deal with choosing types for the particular domain and grouping them if itâĂŹs needed. Then we explain the process of transforming structured data from first section, to datasets that are ready to be used in Stanford NER application for training models. And at the end we cover the process of training models that were used in experiments.

Finally we provide all experiments, that were needed to check if domain specific models will provide better results compared to a global models. We cover the highlighted goals of the experiments. Afterwards the evaluation metric are explained the evaluation metrics used to compare results from experiments. And finally, we go through all provided experiments that were needed to answer to the set goals. One main experiment was created where all other experiments were compared with those results. As well we have used our biggest trained models and test them with web articles from BCC and CNN web page.

From the provided experiments we can conclude that creating and training a domain specific models give better results than a global domain that are used today. Another advantage of using a domain specific model is the time. For training and testing those models, is needed less time and less memory. This

knowledge gives an opportunity to train a bigger domain specific models where can be covered more data can be included. As well from the observation on results, the fine grain models provide a bit better results, than a coarse grain model. Disadvantage of these types of model is that is needed more time and memory to train it. But, like advantage is the list of used and recognized entities.

Future work

In the future for providing a even better results than now can be used technique for annotating entities more than once. For example if in text word "Barack Obama" appears more than ones, also other appearance to be annotated. This can brings to have more entities in a lower dataset.

As well adding a new future or flag in the process of training models can also brings for a bigger precision while performing tests.

To transfer and prepare datasets more quickly using another framework than Apache Jena can lower the processing time. Also importing the whole data to some database, for example Virtuoso, and querying data from there maybe will have an impact on processing time.

Bibliography

[1] Named Entity Recognition. Named Entity Recognition NER. Available from: https://en.wikipedia.org/wiki/Named-entity_recognition

[2] Michal Konkol. *Named Entity Recognition*. Master's thesis, University of West Bohemia in Pilsen, https://www.kiv.zcu.cz/site/documents/verejne/vyzkum/publikace/technicke-zpravy/2012/tr-2012-04.pdf, 2012.

[3] Wikipedia. Information extraction IE. Available from: https://en.wikipedia.org/wiki/Information_extraction

[4] Charles Sutton and Andrew McCallum. An Introduction to Conditional Random Fields for Relational Learning. *Introduction to Statistical Relational Learning. Edited by Lise Getoor and Ben Taskar*, 2006. Available from: http://people.cs.umass.edu/~mccallum/papers/crf-tutorial.pdf

[5] Charles Sutton and Andrew McCallum. An Introduction to Conditional Random Fields for Relational Learning. *Introduction to Statistical Relational Learning. Edited by Lise Getoor and Ben Taskar*, 2006. Available from: http://people.cs.umass.edu/~mccallum/papers/crf-tutorial.pdf

[6] Jenny Rose Finkel, Trond Grenager, and Christopher Manning. Incorporating Non-local Information into Information Extraction Systems by Gibbs Sampling. *Proceedings of the 43nd Annual Meeting of the Association for Computational Linguistics (ACL 2005)*, 2005: pp. 363–370. Available from: http://nlp.stanford.edu/~manning/papers/gibbscrf3.pdf

[7] Wikipedia. DBpedia Spotlight. Available from: https://en.wikipedia.org/wiki/DBpedia#DBpedia_Spotlight

[8] Wikipedia. spaCy. Available from: https://en.wikipedia.org/wiki/SpaCy

[9] Wikipedia. GATE. Available from: https://en.wikipedia.org/wiki/General_Architecture_for_Text_Engineering

[10] Wikipedia. Resource Description Framework RDF. Available from: https://en.wikipedia.org/wiki/Resource_Description_Framework

[11] Usmanov Radmir. *NCollection, Transformation, and Integration of Data from the Web Services Domain*. Master's thesis, Czech Technical University in Prague, Faculty of Information Technology, https://dspace.cvut.cz/bitstream/handle/10467/72987/F8-DP-2017-Usmanov-Radmir-thesis.pdf, 2017.

[12] W3C. Resource Description Framework RDF. Available from: https://www.w3.org/RDF/

[13] W3C. Natural Language Processing Interchange Format NIF. Available from: https://www.w3.org/2015/09/bpmlod-reports/nif-based-nlp-webservices/#natural-language-processing-interchange-format-nif

[14] Hellmann, S.; Lehmann, J.; et al. Integrating NLP using Linked Data. 2013. Available from: http://svn.aksw.org/papers/2013/ISWC_NIF/public.pdf

[15] DBpedia. DBpedia Core. Available from: https://wiki.dbpedia.org/dbpedia-wiki

[16] DBpedia. DBpedia NIF Dataset. Available from: http://wiki.dbpedia.org/dbpedia-nif-dataset

[17] DBpedia. DBpedia Ontology. Available from: http://wiki.dbpedia.org/services-resources/ontology

[18] Wikipedia. Apache Jena. Available from: https://en.wikipedia.org/wiki/Apache_Jena

[19] Vivek Kulkarni and Yashar Mehdad and Troy Chevalier. Domain Adaptation for Named Entity Recognition in Online Media with Word Embeddings. *CoRR*, volume abs/1612.00148, 2016, 1612.00148. Available from: http://arxiv.org/abs/1612.00148

[20] Alfonseca, E.; Manandhar, S. An Unsupervised Method for General Named Entity Recognition And Automated Concept Discovery. 2002. Available from: http://www-users.cs.york.ac.uk/~suresh/papers/AUMFGNERAACD.pdf

[21] for Named Entity Recognition in Online Media with Word Embeddings, D. A. Domain Adaptation for Named Entity Recognition in Online Media with Word Embeddings. 2016. Available from: https://arxiv.org/pdf/1612.00148.pdf

[22] Ritter, A.; Clark, S.; et al. Named Entity Recognition in Tweets: An Experimental Study. 2011: pp. 1524–1534. Available from: http://dblp.uni-trier.de/db/conf/emnlp/emnlp2011.html#RitterCME11

[23] Javier D. FernÃąndez and Miguel A. MartÃŋnez-Prieto and Claudio GutiÃĺrrez and Axel Polleres and Mario Arias. Binary RDF Representation for Publication and Exchange (HDT). *Web Semantics: Science, Services and Agents on the World Wide Web*, volume 19, 2013: pp. 22–41. Available from: http://www.websemanticsjournal.org/index.php/ps/article/view/328

[24] Wikipedia. PageRank. Available from: https://en.wikipedia.org/wiki/PageRank

[25] Wikipedia. F1 score. Available from: https://en.wikipedia.org/wiki/F1_score

Appendix A

Appendix

A.1 Acronyms

NER Named-Entity Recognition
NLP Natural Language Processing
RDF Resource Description Framework
NIF Natural Language Processing Interchange Format
IE Information Extraction
CRF Conditional Random Field
GATE General Architecture for Text Engineering
ANNIE A Nearly-New Information Extraction System .

A.2 POLITICS domain types

Types: Politician, Ambassador, Chancellor, Congressman, Deputy, Governor, Lieutenant, Mayor, MemberOfParliament, Minister, President, PrimeMinister, Senator, VicePresident and VicePrimeMinister **are grouped at** Politician type.

Types: Parliament, Election,PoliticalParty, GeopoliticalOrganisation, PoliticianSpouse, PersonFunction, PoliticalFunction, Profession, TopicalConcept and PoliticalConcept **are not grouped**.

A.3 SPORT domain types

Types: Sport, firstOlympicEvent, footedness, TeamSport, SportsClub, HockeyClub, RugbyClub, SoccerClub, chairmanTitle, clubsRecordGoalscorer, fansgroup, firstGame, ground, largestWin, managerTitle, worstDefeat and NationalSoccerClub **are grouped at** SportsClub type.

Types: SportsLeague, AmericanFootballLeague, AustralianFootballLeague, AutoRacingLeague, BaseballLeague, BasketballLeague, BowlingLeague, Box-

ingLeague, CanadianFootballLeague, CricketLeague, CurlingLeague, CyclingLeague, FieldHockeyLeague, FormulaOneRacing, GolfLeague, HandballLeague, IceHockeyLeague, InlineHockeyLeague, LacrosseLeague, MixedMartialArtsLeague, MotorcycleRacingLeague, PaintballLeague, PoloLeague, RadioControlledRacingLeague, RugbyLeague, SoccerLeague, SoftballLeague, SpeedwayLeague, TennisLeague, VideogamesLeague and VolleyballLeague **are grouped at** SportsLeague type.

Types: SportsTeam, AmericanFootballTeam, AustralianFootballTeam, BaseballTeam, BasketballTeam, CanadianFootballTeam, CricketTeam, CyclingTeam, FormulaOneTeam, HandballTeam, HockeyTeam and SpeedwayTeam **are grouped at** SportsTeam type.

Types: Athlete, ArcherPlayer, AthleticsPlayer, AustralianRulesFootballPlayer, BadmintonPlayer, BaseballPlayer, BasketballPlayer, Bodybuilder, Boxer, AmateurBoxer, BullFighter, Canoeist, ChessPlayer, Cricketer, Cyclist, DartsPlayer, Fencer, GaelicGamesPlayer, GolfPlayer, GridironFootballPlayer, AmericanFootballPlayer, CanadianFootballPlayer, Gymnast, HandballPlayer, HighDiver, HorseRider, Jockey, LacrossePlayer, MartialArtist, MotorsportRacer, MotorcycleRider, MotocycleRacer, SpeedwayRider, RacingDriver, DTMRacer, FormulaOneRacer, NascarDriver, RallyDriver, NationalCollegiateAthleticAssociationAthlete, NetballPlayer, PokerPlayer, Rower, RugbyPlayer, SnookerPlayer, SnookerChamp, SoccerPlayer, SquashPlayer, Surfer, Swimmer, TableTennisPlayer, TeamMember, TennisPlayer, VolleyballPlayer, BeachVolleyballPlayer , WaterPoloPlayer, WinterSportPlayer, Biathlete, BobsleighAthlete , CrossCountrySkier, Curler, FigureSkater, IceHockeyPlayer, NordicCombined, Skater, Ski_jumper, Skier, SpeedSkater, Wrestler, SumoWrestler, Athletics and currentWorldChampion **are grouped at** Athlete type.

Types: Coach, AmericanFootballCoach, CollegeCoach and VolleyballCoach **are grouped at** Coach type.

Types: OrganizationMember, SportsTeamMember **are grouped at** OrganizationMember type.

Types: SportsManager, SoccerManager **are grouped at** SportsManager type.

Types: SportsEvent, CyclingCompetition, FootballMatch, GrandPrix, InternationalFootballLeagueEvent, MixedMartialArtsEvent, NationalFootballLeagueEvent, Olympics, OlympicEvent, Race, CyclingRace, HorseRace, MotorRace, Tournament, GolfTournament, SoccerTournament, TennisTournament, WomensTennisAssociationTournament, WrestlingEvent, SportCompetitionResult, OlympicResult, SnookerWorldRanking, SportsSeason, MotorsportSeason, SportsTeamSeason, BaseballSeason, FootballLeagueSeason, NationalFootballLeagueSeason, NCAATeamSeason, SoccerClubSeason, SoccerLeagueSeason and MotorSportSeason **are grouped at** SportsEvent type.

A.4 TRANSPORTATION domain types

Types: Aircraft, aircraftType, aircraftUser, ceiling, dischargeAverage, enginePower, engineType, gun, powerType, wingArea, wingspan and MilitaryAircraft **are grouped at** Aircraft type.

Types: Automobile, automobilePlatform, bodyStyle, transmission and AutomobileEngine **are grouped at** Automobile.

Types: Locomotive, boiler and CylinderCount **are grouped at** Locomotive.

Types: MilitaryVehicle, Motorcycle, SpaceStation **are not grouped.**

Types: On-SiteTransportation, ConveyorSystem, Escalator and MovingWalkway **are grouped at** On-SiteTransportation type.

Types: Rocket, countryOrigin, finalFlight, lowerEarthOrbitPayload, maidenFlight, rocketFunction, rocketStages and RocketEngine **are grouped at** Rocket type.

Types: Ship, captureDate, homeport, layingDown, maidenVoyage, numberOfPassengers, shipCrew and shipLaunch **are grouped at** Ship type.

Types: SpaceShuttle, contractAward, Crews, firstFlight, lastFlight, missions, numberOfCrew, numberOfLaunches and satellitesDeployed **are grouped at** SpaceShuttle type.

Types: Spacecraft, cargoFuel, cargoGas, cargoWater and rocket **are grouped at** Spacecraft type.

Types: Train, locomotive, wagon and TrainCarriage **are grouped at** Train type.

Types: Tram, PublicTransitSystem, Airline and BusCompany **are grouped at** PublicTransitSystem type.

Types: Infrastructure, Airport, Port, RestArea, RouteOfTransportation, Bridge, RailwayLine, RailwayTunnel, WaterwayTunnel, Station, MetroStation, RailwayStation, RouteStop and TramStation **are grouped at** Infrastructure type.

A.5 Properties file used for training models

```
# location of the training file
trainFile =
# location where you would like to save
#(serialize) your
# classifier; adding .gz at the end
#automatically gzips the file,
# making it smaller, and faster to load
serializeTo =

# structure of your training file;
```

```
# this tells the classifier that
# the word is in column 0 and the
# correct answer is in column 1
map = word=0,answer=1

# This specifies the order of the CRF:
# order 1 means that features
# apply at most to a class pair of
# previous class and current class
# or current class and next class.
maxLeft=1

# these are the features we'd like to
# train with
# some are discussed below, the rest can
# be understood by looking
# at NERFeatureFactory
useClassFeature=true
useWord=true
# word character ngrams will be included
# up to length 6 as prefixes
# and suffixes only
useNGrams=true
noMidNGrams=true
maxNGramLeng=6
usePrev=true
useNext=true
useDisjunctive=true
useSequences=true
usePrevSequences=true
saveFeatureIndexToDisk=true
useObservedSequencesOnly=true
# the last 4 properties deal with word
# shape features
useTypeSeqs=true
useTypeSeqs2=true
useTypeySequences=true
wordShape=chris2useLC
```

A.6 POLITICS domain SPARQL query

```
PREFIX rdf:<http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX dbo:<http://dbpedia.org/ontology/>
```

```
PREFIX vrank:<http://purl.org/voc/vrank#>

SELECT ?s ?v
FROM <http://dbpedia.org>
FROM <http://people.aifb.kit.edu/ath/#DBpedia_PageRank>
WHERE {
{?s rdf:type dbo:PoliticalParty .}
UNION
{?s rdf:type dbo:Politician .}
UNION
{
        ?s rdf:type dbo:SocietalEvent.
        ?s rdf:type dbo:Election.
}
UNION
{
        ?s rdf:type dbo:Person.
        ?s rdf:type dbo:OfficeHolder.
}
?s vrank:hasRank/vrank:rankValue ?v.
}
ORDER BY DESC(?v) LIMIT 10
```

A.7 SPORT domain SPARQL query

```
PREFIX rdf:<http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX dbo:<http://dbpedia.org/ontology/>

PREFIX vrank:<http://purl.org/voc/vrank#>

SELECT ?s ?v
FROM <http://dbpedia.org>
FROM <http://people.aifb.kit.edu/ath/#DBpedia_PageRank>
WHERE {
{?s rdf:type dbo:Sport.}
UNION
{?s rdf:type dbo:SportsLeague.}
UNION
{?s rdf:type dbo:SportsTeam.}
UNION
{?s rdf:type dbo:Athlete.}
UNION
{?s rdf:type dbo:AustralianRulesFootballPlayer.}
UNION
```

```
{?s rdf:type dbo:BadmintonPlayer.}
UNION
{?s rdf:type dbo:BaseballPlayer.}
UNION
{?s rdf:type dbo:BasketballPlayer.}
UNION
{?s rdf:type dbo:Bodybuilder.}
UNION
{?s rdf:type dbo:Boxer.}
UNION
{?s rdf:type dbo:Canoeist.}
UNION
{?s rdf:type dbo:ChessPlayer.}
UNION
{?s rdf:type dbo:Cricketer.}
UNION
{?s rdf:type dbo:Cyclist.}
UNION
{?s rdf:type dbo:DartsPlayer.}
UNION
{?s rdf:type dbo:Fencer.}
UNION
{?s rdf:type dbo:GaelicGamesPlayer.}
UNION
{?s rdf:type dbo:GolfPlayer.}
UNION
{?s rdf:type dbo:GridironFootballPlayer.}
UNION
{?s rdf:type dbo:Gymnast.}
UNION
{?s rdf:type dbo:HandballPlayer.}
UNION
{?s rdf:type dbo:HighDiver.}
UNION
{?s rdf:type dbo:HorseRider.}
UNION
{?s rdf:type dbo:Jockey.}
UNION
{?s rdf:type dbo:LacrossePlayer.}
UNION
{?s rdf:type dbo:MartialArtist.}
UNION
{?s rdf:type dbo:MotorsportRacer.}
UNION
```

```
{?s rdf:type dbo:NationalCollegiateAthleticAssociationAthlete .}
UNION
{?s rdf:type dbo:NetballPlayer.}
UNION
{?s rdf:type dbo:PokerPlayer.}
UNION
{?s rdf:type dbo:RugbyPlayer.}
UNION
{?s rdf:type dbo:SnookerPlayer.}
UNION
{?s rdf:type dbo:SoccerPlayer.}
UNION
{?s rdf:type dbo:SquashPlayer.}
UNION
{?s rdf:type dbo:Surfer.}
UNION
{?s rdf:type dbo:Swimmer.}
UNION
{?s rdf:type dbo:TableTennisPlayer.}
UNION
{?s rdf:type dbo:TeamMember.}
UNION
{?s rdf:type dbo:TennisPlayer.}
UNION
{?s rdf:type dbo:VolleyballPlayer.}
UNION
{?s rdf:type dbo:WaterPoloPlayer.}
UNION
{?s rdf:type dbo:WinterSportPlayer.}
UNION
{?s rdf:type dbo:Wrestler.}
UNION
{?s rdf:type dbo:Coach.}
UNION
{?s rdf:type dbo:OrganisationMember.}
UNION
{?s rdf:type dbo:SportsManager.}
UNION
{?s rdf:type dbo:SportsEvent.}
UNION
{?s rdf:type dbo:Race.}
UNION
{?s rdf:type dbo:Tournament.}
UNION
```

```
{?s rdf:type dbo:WrestlingEvent.}
UNION
{?s rdf:type dbo:SportCompetitionResult.}
UNION
{?s rdf:type dbo:SportsSeason.}
UNION
{?s rdf:type dbo:Referee.}
UNION
{?s rdf:type dbo:SportFacility.}
UNION
{?s rdf:type dbo:CricketGround.}
UNION
{?s rdf:type dbo:GolfCourse.}
UNION
{?s rdf:type dbo:RaceTrack.}
UNION
{?s rdf:type dbo:SkiArea.}

?s vrank:hasRank/vrank:rankValue ?v.
}
ORDER BY DESC(?v) LIMIT 10
```

A.8 TRANSPORTATION domain SPARQL query

```
PREFIX rdf:<http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX dbo:<http://dbpedia.org/ontology/>

PREFIX vrank:<http://purl.org/voc/vrank#>

SELECT ?s ?v
FROM <http://dbpedia.org>
FROM <http://people.aifb.kit.edu/ath/#DBpedia_PageRank>
WHERE {
{?s rdf:type dbo:Aircraft.}
UNION
{?s rdf:type dbo:Automobile.}
UNION
{?s rdf:type dbo:Locomotive.}
UNION
{?s rdf:type dbo:MilitaryVehicle.}
UNION
{?s rdf:type dbo:Motorcycle.}
UNION
{?s rdf:type dbo:On-SiteTransportation.}
```

```
UNION
{?s rdf:type dbo:Rocket.}
UNION
{?s rdf:type dbo:Ship.}
UNION
{?s rdf:type dbo:SpaceShuttle.}
UNION
{?s rdf:type dbo:SpaceStation.}
UNION
{?s rdf:type dbo:Spacecraft.}
UNION
{?s rdf:type dbo:Train.}
UNION
{?s rdf:type dbo:TrainCarriage.}
UNION
{?s rdf:type dbo:Tram.}
UNION
{?s rdf:type dbo:Engine.}
UNION
{?s rdf:type dbo:AutomobileEngine.}
UNION
{?s rdf:type dbo:RocketEngine.}
UNION
{
?s rdf:type dbo:Company.
?s rdf:type dbo:PublicTransitSystem.
}
UNION
{?s rdf:type dbo:Airline.}
UNION
{?s rdf:type dbo:BusCompany.}
UNION
{?s rdf:type dbo:Infrastructure.}
UNION
{?s rdf:type dbo:Airport.}
UNION
{?s rdf:type dbo:Port.}
UNION
{?s rdf:type dbo:RestArea.}
UNION
{?s rdf:type dbo:RouteOfTransportation.}
UNION
{?s rdf:type dbo:Bridge.}
UNION
```

```
{?s rdf:type dbo:RailwayLine.}
UNION
{?s rdf:type dbo:RailwayTunnel.}
UNION
{?s rdf:type dbo:Road.}
UNION
{?s rdf:type dbo:RoadJunction.}
UNION
{?s rdf:type dbo:RoadTunnel.}
UNION
{?s rdf:type dbo:WaterwayTunnel.}
UNION
{?s rdf:type dbo:Station.}
UNION
{?s rdf:type dbo:MetroStation.}
UNION
{?s rdf:type dbo:RailwayStation.}
UNION
{?s rdf:type dbo:RouteStop.}
UNION
{?s rdf:type dbo:TramStop.}


?s vrank:hasRank/vrank:rankValue ?v.
}
ORDER BY DESC(?v) LIMIT 10
```

Appendix B

Contents of CD

- `readme.txt` ..... The file with CD contents description
- `CreateModels` ..... Thesis project source code
  - `src` ..... The directory of source codes
  - `pom.xml` ..... Maven Project Object Model
- `Stanford NER` Directory with Stanford NER application, models and test datasets
  - `classifiers` ..... Stanford NER classifiers
  - `domainsWithOneAnnotation` . The directory that contains test dataset and properties files used for training models
  - `edu` ..... The directory that contains Stanford NER source codes
  - `lib` ..... The directory that contains library used in Stanford NER application
  - `META-INF` ..... The directory that contains Stanford NER metadata
  - `modelsWithOneAnnotation`. Directory contains trained models used in experiments
- `Text` ..... The thesis text directory
  - `Bogoljub_Jakovcheski_Master_Thesis_2018.pdf` . The diploma thesis in PDF format
- `Thesis LaTeX` ..... the directory of LaTeX source codes of the thesis