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 7, 2018

Acknowledgements

I would like to thank my family and friends 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 7, 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

Klíčová slova

Abstract

Keywords

Contents

Citation of this thesis vi

Introduction **1**
Motivation 1
Goals of the thesis 2
Thesis outline 2

1 Background and related work **3**
1.1 Background 3
1.1.1 Information extraction 3
1.1.2 Named Entity Recognition 4
1.1.3 RDF/NIF 7
1.1.4 DBpedia 7
1.1.5 Apache Jena 9
1.1.6 SPARQL 9
1.2 Related work 10
1.2.1 Domain specific Named Entity Recognition 10

2 Domain specific named entity recognition **11**
2.1 Data pre-processing 11
2.2 Domain specification 13
2.3 Types retrieval 13
2.4 Data transformation 15
2.5 Model generation 17
2.5.1 Training datasets 17

3 Experiments **19**
3.1 F_1 score 19
3.2 Goals of the experiments 19
3.3 List of experiments 20
3.3.1 Main experiment 20

3.3.2 Experiments that has less than 300 abstracts in model . 27
3.3.3 Experiments that have more than 300 abstracts in model and test files 42
3.3.4 MIXED 51
3.3.5 Experiments with lower training abstracts on model, but higher abstracts on test file 54
3.3.6 Experiments with higher training abstracts on model, but lower abstracts on test file 62
3.4 Summary of results 73
3.4.1 Graphs 73

Conclusion **75**

Bibliography **77**

A Retrieved types **81**
A.1 POLITICS types 81
A.2 SPORT types 81
A.3 TRANSPORTATION types 82

B Stanford NER properties file **85**

C Contents of CD **87**

List of Figures

1.1 Information extraction example 4
1.2 Stanford NER GUI with 3 classes model (Location, Person, Organization) 5
1.3 Dbpedia Ontology - Instances per class 9

3.1 All 3 Domains Fine Grained Top 10 Links Runned With All 3 Domains Fine Grained Top 100 Links 54
3.2 All 3 Domains Fine Grained Top 10 Links Runned With All 3 Domains Fine Grained Top 300 Links 55
3.3 All 3 Domains Fine Grained Top 10 Links Runned With All 3 Domains Fine Grained Top 500 Links 55
3.4 All 3 Domains Fine Grained Top 10 Links Runned With All 3 Domains Fine Grained Top 500 Links With Lower PageRank . . . 56
3.5 All 3 Domains Fine Grained Top 100 Links Runned With All 3 Domains Fine Grained Top 500 Links 56
3.6 All 3 Domains Fine Grained Top 100 Links Runned With All 3 Domains Fine Grained Top 500 Links With Lower PageRank . . . 57
3.7 All 3 Domains Fine Grained Top 100 Links Runned With Politics Fine Grained Top 500 Links 57
3.8 All 3 Domains Fine Grained Top 100 Links Runned With Politics Fine Grained Top 500 Links With Lower PageRank 57
3.9 All 3 Domains Fine Grained Top 100 Links Runned With Sport Fine Grained Top 500 Links 58
3.10 All 3 Domains Fine Grained Top 100 Links Runned With Sport Fine Grained Top 500 Links With Lower PageRank 58
3.11 All 3 Domains Fine Grained Top 100 Links Runned With Transportation Fine Grained Top 500 Links 58
3.12 All 3 Domains Fine Grained Top 100 Links Runned With Transportation Fine Grained Top 500 Links With Lower PageRank . . . 59

3.13 All 3 Domains Fine Grained Top 300 Links Runned With All 3 Domains Fine Grained Top 500 Links 59
3.14 All 3 Domains Fine Grained Top 300 Links Runned With All 3 Domains Fine Grained Top 500 Links With Lower PageRank . . . 60
3.15 All 3 Domains Fine Grained Top 300 Links Runned With Politics Fine Grained Top 500 Links 60
3.16 All 3 Domains Fine Grained Top 300 Links Runned With Politics Fine Grained Top 500 Links With Lower PageRank 60
3.17 All 3 Domains Fine Grained Top 300 Links Runned With Sport Fine Grained Top 500 Links 61
3.18 All 3 Domains Fine Grained Top 300 Links Runned With Sport Fine Grained Top 500 Links With Lower PageRank 61
3.19 All 3 Domains Fine Grained Top 300 Links Runned With Transportation Fine Grained Top 500 Links 61
3.20 All 3 Domains Fine Grained Top 300 Links Runned With Transportation Fine Grained Top 500 Links With Lower PageRank . . . 62
3.21 All 3 Domains Fine Grained Top 300 Links Runned With All 3 Domains Fine Grained Top 10 Links 62
3.22 All 3 Domains Fine Grained Top 300 Links Runned With All 3 Domains Fine Grained Top 100 Links 63
3.23 All 3 Domains Fine Grained Top 300 Links Runned With Politics Fine Grained Top 10 Links 63
3.24 All 3 Domains Fine Grained Top 300 Links Runned With Politics Fine Grained Top 100 Links 63
3.25 All 3 Domains Fine Grained Top 300 Links Runned With Sport Fine Grained Top 10 Links 64
3.26 All 3 Domains Fine Grained Top 300 Links Runned With Sport Fine Grained Top 100 Links 64
3.27 All 3 Domains Fine Grained Top 300 Links Runned With Transportation Fine Grained Top 10 Links 64
3.28 All 3 Domains Fine Grained Top 300 Links Runned With Transportation Fine Grained Top 100 Links 65
3.29 All 3 Domains Fine Grained Top 500 Links Runned With All 3 Domains Fine Grained Top 10 Links 65
3.30 All 3 Domains Fine Grained Top 500 Links Runned With All 3 Domains Fine Grained Top 100 Links 65
3.31 All 3 Domains Fine Grained Top 500 Links Runned With All 3 Domains Fine Grained Top 300 Links 66
3.32 All 3 Domains Fine Grained Top 500 Links Runned With Politics Fine Grained Top 10 Links 66
3.33 All 3 Domains Fine Grained Top 500 Links Runned With Politics Fine Grained Top 100 Links 66
3.34 All 3 Domains Fine Grained Top 500 Links Runned With Politics Fine Grained Top 300 Links 67

3.35 All 3 Domains Fine Grained Top 500 Links Runned With Sport Fine Grained Top 10 Links 67
3.36 All 3 Domains Fine Grained Top 500 Links Runned With Sport Fine Grained Top 100 Links 67
3.37 All 3 Domains Fine Grained Top 500 Links Runned With Sport Fine Grained Top 300 Links 68
3.38 All 3 Domains Fine Grained Top 500 Links Runned With Transportation Fine Grained Top 10 Links 68
3.39 All 3 Domains Fine Grained Top 500 Links Runned With Transportation Fine Grained Top 100 Links 68
3.40 All 3 Domains Fine Grained Top 500 Links Runned With Transportation Fine Grained Top 300 Links 69
3.41 Politics Fine Grained Top 300 Links Runned With All 3 Domains Fine Grained Top 100 Links 69
3.42 Politics Fine Grained Top 500 Links Runned With All 3 Domains Fine Grained Top 100 Links 70
3.43 Politics Fine Grained Top 500 Links Runned With All 3 Domains Fine Grained Top 300 Links 70
3.44 Sport Fine Grained Top 300 Links Runned With All 3 Domains Fine Grained Top 100 Links 71
3.45 Sport Fine Grained Top 500 Links Runned With All 3 Domains Fine Grained Top 100 Links 71
3.46 Sport Fine Grained Top 500 Links Runned With All 3 Domains Fine Grained Top 300 Links 72
3.47 Transportation Fine Grained Top 300 Links Runned With All 3 Domains Fine Grained Top 100 Links 72
3.48 Transportation Fine Grained Top 500 Links Runned With All 3 Domains Fine Grained Top 100 Links 73
3.49 Transportation Fine Grained Top 500 Links Runned With All 3 Domains Fine Grained Top 300 Links 73

List of Tables

3.1 Testing computer parameters 19
3.2 Outcomes of base experiment run to be used as reference for subsequential experiments 20
3.3 Outcomes of base model in coarse grained run with ”POLITICS” abstracts 21
3.4 Outcomes of base model in coarse grained run with ”SPORT” abstracts 21
3.5 Outcomes of base model in coarse grained run with ”TRANSPORTATION” abstracts 21
3.6 Outcomes of base experiment in fine grained run to be used as reference for subsequential experiments 22
3.7 Outcomes of base model in fine grained run with ”POLITICS” abstracts 23
3.8 Outcomes of base model in fine grained run with ”SPORT” abstracts 23
3.9 Outcomes of base model in fine grained run with ”TRANSPORTATION” abstracts 24
3.10 Outcomes of ”POLITICS” base model in coarse grained run with ”POLITICS” abstracts 25
3.11 Outcomes of ”POLITICS” base model in fine grained run with ”POLITICS” abstracts 25
3.12 Outcomes of ”SPORT” base model in coarse grained run with ”SPORT” abstracts 25
3.13 Outcomes of ”SPORT” base model in fine grained run with ”SPORT” abstracts 26
3.14 Outcomes of ”TRANSPORTATION” base model in coarse grained run with ”TRANSPORTATION” abstracts 26
3.15 Outcomes of ”TRANSPORTATION” base model in fine grained run with ”TRANSPORTATION” abstracts 27

3.16 Outcomes of global model in coarse grained run with 10 abstracts from every domain 28
3.17 Outcomes of global model in coarse grained run with 10 abstracts from ”POLITICS” domain 28
3.18 Outcomes of global model in coarse grained run with 10 abstracts from ”SPORT” domain 28
3.19 Outcomes of global model in coarse grained run with 10 abstracts from ”TRANSPORTATION” domain 29
3.20 Outcomes of global model in fine grained run with 10 abstracts from every domain 30
3.21 Outcomes of global model in fine grained run with 10 abstracts from ”POLITICS” domain 30
3.22 Outcomes of global model in fine grained run with 10 abstracts from ”SPORT” domain 31
3.23 Outcomes of global model in fine grained run with 10 abstracts from ”TRANSPORTATION” domain 31
3.24 Outcome of ”POLITICS” domain specific model in coarse grained run with 10 abstracts from the same domain 32
3.25 Outcome of ”POLITICS” domain specific model in fine grained run with 10 abstracts from the same domain 32
3.26 TABLE 32
3.27 TABLE 32
3.28 TABLE 33
3.29 TABLE 33
3.30 TABLE 33
3.31 TABLE 33
3.32 TABLE 33
3.33 TABLE 33
3.34 TABLE 34
3.35 TABLE 34
3.36 TABLE 34
3.37 TABLE 35
3.38 TABLE 35
3.39 TABLE 35
3.40 TABLE 35
3.41 TABLE 35
3.42 TABLE 36
3.43 TABLE 36
3.44 TABLE 36
3.45 TABLE 36
3.46 TABLE 36
3.47 TABLE 36
3.48 TABLE 37
3.49 TABLE 37

3.50 TABLE 37
3.51 TABLE 38
3.52 TABLE 38
3.53 TABLE 38
3.54 TABLE 38
3.55 TABLE 38
3.56 TABLE 39
3.57 TABLE 39
3.58 TABLE 39
3.59 TABLE 39
3.60 TABLE 39
3.61 TABLE 39
3.62 TABLE 40
3.63 TABLE 40
3.64 TABLE 40
3.65 TABLE 41
3.66 TABLE 41
3.67 TABLE 41
3.68 TABLE 41
3.69 TABLE 42
3.70 TABLE 42
3.71 TABLE 42
3.72 TABLE 42
3.73 TABLE 43
3.74 TABLE 43
3.75 TABLE 43
3.76 TABLE 44
3.77 TABLE 44
3.78 TABLE 45
3.79 TABLE 45
3.80 TABLE 45
3.81 TABLE 46
3.82 TABLE 46
3.83 TABLE 46
3.84 TABLE 46
3.85 TABLE 46
3.86 TABLE 47
3.87 TABLE 47
3.88 TABLE 47
3.89 TABLE 47
3.90 TABLE 48
3.91 TABLE 48
3.92 TABLE 49
3.93 TABLE 49

3.94 TABLE 49
3.95 TABLE 50
3.96 TABLE 50
3.97 TABLE 50
3.98 TABLE 50
3.99 TABLE 50
3.100All 3 Domains Fine Grained Top 300 With All 3 Domains Fine Grained Top 500 Links And All 3 Domains Fine Grained Top 500 Links With Lower PageRank 51
3.101All 3 Domains Fine Grained Top 500 Links With All 3 Domains Fine Grained Top 500 Links And All 3 Domains Fine Grained Top 500 Links With Lower Page Rank 52
3.102All 3 Domains Fine Grained Top 500 Links With All 3 Domains Fine Grained Top 500 Links With Lower PageRank 53
3.103Transportation Fine Grained Top 500 Links Runned With All 3 Domains Fine Grained Top 300 Links And Transportation Fine Grained Top 300 Links 54

Introduction

Motivation

Named Entity Recognition (NER)[1] is 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 NLP tasks [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 nice results on texts or domains that are trained, but bad results for texts on a specific domain for which that NER 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. 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 created with a certain number on types from DBpedia Ontology, then for creating datasets is used DBpedia NIF who gives an opportunity to approaches to information from Wikipedia abstracts, for example, types that annotated words has in those abstracts.

Thesis research which is the quality of trained domains, the impact of the size of the data and the quality of the defined domains.

[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

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:

- 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.

Thesis outline

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. Unlike linear-chain models, general CRFs can capture long distance dependencies between labels [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 information from mail message and adding to your Calendar 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 **??** 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/

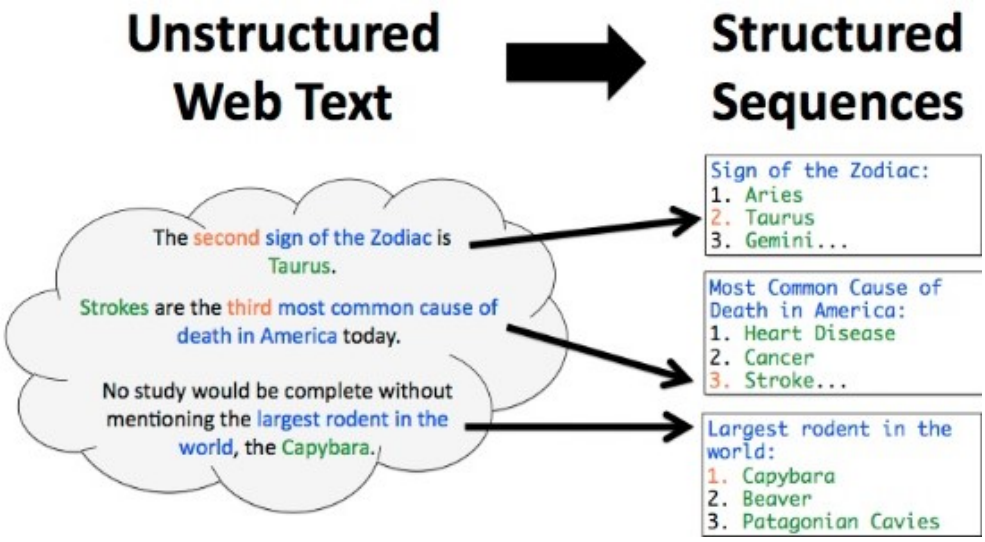

Figure 1.1: Information extraction, downloaded from[6]

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.

One approach to NER is to classify each word independently as one of either Person, Location, Organization, or Other (meaning not an entity). The problem with this approach is that it assumes that given the input, all of the named entity labels are independent. In fact, the named-entity labels of neighboring words are dependent; for example, while New York is a location, New York Times is an organization.

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.

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

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

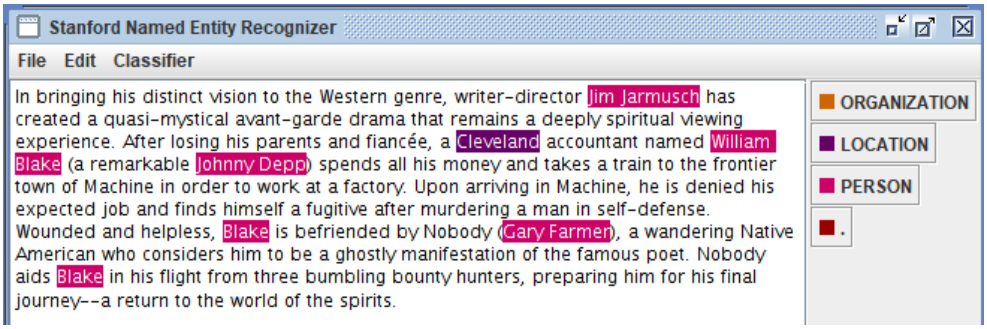

Figure 1.2: Stanford NER GUI with 3 classes model (Location, Person, Organization)

There are several applications or frameworks for NER like Stanford NER, DBpedia Spotlight, spaCy, Chatbot NER, GATE, OpenNLP and so on. 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. Included with the download are good named entity recognizers for English, particularly for the 3 classes (PERSON, ORGANIZATION, LOCATION), and we also make available on this page various other models for different languages and circumstances, including models trained on just the CoNLL 2003 English training data.

Stanford NER is also known 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].

[7] https://nlp.stanford.edu/software/CRF-NER.html#Models
[8] https://nlp.stanford.edu/software/CRF-NER.html

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 Open Data cloud through DBpedia. DBpedia Spotlight performs named entity extraction, including entity detection and name resolution (in other words, disambiguation). It can also be 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.

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

[9] https://www.dbpedia-spotlight.org/
[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

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. 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/NIF

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 stored in a triplestore [11].

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.

Natural Language Processing Interchange Format (NIF)[15] [13] is an RDF-based format. The classes to represent linguistic data are defined in the NIF Core Ontology. All ontology classes are derived from the main class nif:String which respresents strings of Unicode characters.

1.1.4 DBpedia

DBpedia [14] 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

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

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 3.7.

1.1.4.1 DBpedia NIF

DBpedia [15] 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. In order to broaden and deepen the amount of structured DBpedia data, we are going a step further.

With the representation of wiki pages in the NLP Interchange Format (NIF) we provide 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

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 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 3.7 release, the ontology is a directed-acyclic graph, not a tree. Classes may have multiple superclasses, which was important for the mappings to schema.org. A taxonomy can still be constructed by ignoring all superclasses except the one that is specified first in the list and is considered the most important [16].

Dbpedia ontology classes can be found here [16]

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

The DBpedia Ontology currently contains about 4,233,000 instances. Figure 1.3 shows the number of instances for several classes within the ontology. [http://wiki.dbpedia.org/services-resources/ontology]

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[17] [17] is an open source Semantic Web framework for Java. It provides an API to extract data from and write to RDF graphs. The graphs 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 [18] 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.

[17] `https://jena.apache.org/index.html`
[18] `https://ontotext.com/knowledgehub/fundamentals/what-is-sparql/`

In Listing 1.1 is an example of SPARQL query where we are selecting 10 abstracts from DBpedia NIF who has ontology type PoliticalParty and their PageRank and sort descending by PageRank.

Listing 1.1: SPARQL example

```
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
```

1.2 Related work

In this section we will compare our approach and our chosen domains.

1.2.1 Domain specific Named Entity Recognition

Traditionally Named Entity Recognition (NER)[18] 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 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.

Chapter 2

Domain specific named entity recognition

In this chapter we will go through the whole process of transforming raw DBpedia datasets to datasets that are ready for training a model with Stanford NER and how to train a model with Stanford NER. Section 2.1 explains the process of cleaning the data 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 how to train a datasets with Stanford NER.

2.1 Data pre-processing

To be able to create domain specific datasets ideally we need some big raw data. 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. From here we needed only 2 datasets, and that nif-context (or nif-abstract-context) and nif-text-links.

Another dataset that we needed was DBpedia instance types dataset, found at DBpedia download page[19] 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 has annotation, but we still don't know their type. So here comes instance types file where based on link from nif-text-link (eg.http://dbpedia.org/resource/Philip_II_of_Macedon) we can find the type

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

of annotated word (word Philip II has ontology type Monarch), but of course, there can 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).

But now, let us explain deeply how we process and clean data from the datasets. First, we define small test dataset to check how fast we can process data. Running that dataset on downloaded files without any cleaning on data takes too long. So we said that converting the datasets from RDF format to binary format (.ttl to .hdt) with RDF/HDT tool[20] will be faster. HDT (Header, Dictionary, Triples)[19] 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 algorithm again. There were some improvements, but not satisfying for our purposes. Our next solution was to clean datasets from 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 mayor memory improvements. Again we rerun the algorithm, of course, there were improvements, but as well as previous the time that algorithm runs, was not acceptable for us. 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 once more we converted our cleaned datasets from RDF format(.ttl) to binary format(.hdt). And how in previous running there were again improvements, but those improvements don't fulfill our expectations. The final thing that we have to save us was creating a dataset tree only for nif-text-links and instance-types files. For nif-text-links file 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 word Volkswagen. With this, we have a smaller dataset where we can read the whole one very fast.

For instance types file we modified the algorithm for creating a data tree. Here because of lower range data we have created only files from "a-z", of course, special characters files and other file.

Finally, we rerun the algorithm, and the time to process one abstract, at

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

worst case, takes no longer than 1 minute. Now we were ready to take next steps to retrieve types (see Section 2.3), create domains (see Section 2.2) 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 this 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.

Ideally will be those domains to have some connection between them and again not to be already widespread. So we look at ontology types that are retrieved for "TRANSPORTATION" domain, and there are types like Airport, Bridge, MetroStation and so on. This indicates to us that next domain can be "POLITICS". Why? Because some airports, bridges or metro stations bear names of Politicians. For instance airport in Prague, Czech Republic is named by the last president of Czechoslovakia, Vaclav Havel. Or another example is that some bridges in the United States are named by famous politicians, like Presidents. 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.

TODO MISSING EXPLANATION OF SPORT DOMAIN!!

2.3 Types retrieval

After we solved the problem of that how effectively run 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[21] 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 on one hand is good, but training the model will take a lot of time and memory, and 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.1, 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. Although this domain has the lowest number of retrieved types, on creating a models this domain has the largest annotated words.

We do the same for "SPORT" domain where we retrieve in total 171 types, found in Appendix A.2, 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.2. 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 further don't lose the sense of type. 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.

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.3. Those types are after minimized in 14 more specific types like Aircraft, Automobile, Locomotive, MilitaryVehicle, Motorcycle, On-SiteTransportation, 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 specific ones is that, that when the dataset has a smaller number of types, training a model with Stenford 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

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

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 web 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. The solution that we choose who fits to our requirements is PageRank. PageRank [20] is an algorithm used by Google Search to rank websites in their search engine results. So with a prepared and tested SPARQL queries on www.dbpedia.com/sparql and with help of Apache Jena framework (see Section 1.1.5) we implemented retrieving links, on Java, on DBpedia endpoint[22]. After retrieving those data, based on their PageRank we search does retrieved link is part on our abstract file. If link is found in nif-abstract 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, 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 a coarse and fine grained files. The algorithm is very simple, it reads the files who already has 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.

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

```
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[23] is a very nice explanation of that how to train own model with Stanford NER. We follow those steps and used pretty much 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. More interesting is useObservedSequencesOnly flag. It's stands 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 B.

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 datasets, coarse-grained and fine-grained, also those 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 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 we have one dataset

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

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. The order of the abstracts is based on PageRank as explained in section 2.4.

3.1 F_1 score

The success of NER systems is exposed to F_1 score (F-score or F-measure). F_1 [21] 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 F1 score is the harmonic average of the precision and recall, where an F1 score reaches its best value at 1 (perfect precision and recall) and worst at 0. Written in formula, the $F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$.

3.2 Goals of the experiments

We set a few goals of the experiments. First of all we waned to test does we will get better results if we run the model of all domains in coarse grained,

against the model of all domains in fine grained. In this test we run the models also with all domains texts. Then we get those models and we run it with specific domain texts, in both fine and coarse grained. Also we make experiments with specific domain model runned with domain specific texts, for example, politics domain model in coarse grained is runned with politics domain text also annotated in coarse grained, politics domain model in fine grained is runned with politics domain text also annotated in fine grained, and the same for sport and transportation domains.

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 like 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.

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 8622805705290 nanoseconds or 2.40 hours. The model is trained in coarse grained and takes 844.63 seconds in optimization and 873.7 seconds on CRFClassifier training.

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 not bad at all, we are above 95% as shown in Table 3.2, which is great number for such middle weight model. With such results, someone will say that those are nice results and other experiments will only have a worst results. But let see how model will behaves when we tested with abstracts for every specific domain.

Entity	**P**	**R**	**F1**
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: Outcomes 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	**P**	**R**	**F1**
POLITICS	0,9839	0,4025	0,5713
TRANSPORTATION	0,0000	1,0000	0,0000
Totals	**0,9792**	**0,4025**	**0,5705**

Table 3.3: Outcomes 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	**P**	**R**	**F1**
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: Outcomes 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	**P**	**R**	**F1**
SPORT	0,0000	1,0000	0,0000
TRANSPORTATION	0,9940	0,9822	0,9880
Totals	**0,9861**	**0,9822**	**0,9841**

Table 3.5: Outcomes 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 waned 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	**P**	**R**	**F1**
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: Outcomes 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	P	R	F1
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: Outcomes 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 this 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	P	R	F1
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: Outcomes 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 contains.

Entity	P	R	F1
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: Outcomes of base model in fine grained run with "TRANSPORTATION" abstracts

Provided experiments with the model who 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 a models trained in coarse grained, everything depends on us. Does we want to trained models faster or we want to be more precise.

3.3.1.2 Specific domain models

After completing experiments with a 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	**P**	**R**	**F1**
POLITICS	0,8039	0,6779	0,7355
Totals	**0,8039**	**0,6779**	**0,7355**

Table 3.10: Outcomes of ”POLITICS” base model in coarse grained run with ”POLITICS” abstracts

We repeat previous experiment, but now everything in fine grained. Time for training this kind of model i 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	**P**	**R**	**F1**
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: Outcomes 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	**P**	**R**	**F1**
SPORT	0,9432	0,8839	0,9126
Totals	**0,9432**	**0,8839**	**0,9126**

Table 3.12: Outcomes 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 and provide an experiment. 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	**P**	**R**	**F1**
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: Outcomes 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	**P**	**R**	**F1**
TRANSPORTATION	0,9583	0,9109	0,9340
Totals	**0,9583**	**0,9109**	**0,9340**

Table 3.14: Outcomes of ”TRANSPORTATION” base model in coarse grained run with ”TRANSPORTATION” abstracts

Finally we make an experiment in fine grained. Total training time 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	P	R	F1
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: Outcomes 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 waned 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. Next 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.

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 648721072970 nanoseconds or 10.81 minutes, which comparing with main experiment, where we need 2.40 hours, is way more faster to prepare data. Of coarse this indicates that training models will also be faster than in main 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. 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	**P**	**R**	**F1**
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: Outcomes of global model in coarse grained run with 10 abstracts from every domain

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	**P**	**R**	**F1**
POLITICS	0,9655	0,3636	0,5283
Totals	**0,9655**	**0,3636**	**0,5283**

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

Then we test our global model with abstracts from ”SPORT” domain. 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	**P**	**R**	**F1**
SPORT	1,0000	1,0000	1,0000
Totals	**1,0000**	**1,0000**	**1,0000**

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

Finally we tested the model with abstract from "TRANSPORTATION" domain. 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	**P**	**R**	**F1**
TRANSPORTATION	1,0000	1,0000	1,0000
Totals	**1,0000**	**1,0000**	**1,0000**

Table 3.19: Outcomes 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.

After we finish the experiments with global models in coarse grained, we waned 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.

Training a fine grained model takes in total 124.7 seconds, from which 120,73 seconds spent in optimization. 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	P	R	F1
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: Outcomes of global model in fine grained run with 10 abstracts from every domain

Then how in previous experiments, we take the global train model and test it with abstracts from every specific domain separately. 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 happens the model will perform pretty mach good recognition.

Entity	P	R	F1
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: Outcomes of global model in fine grained run with 10 abstracts from ”POLITICS” domain

Then we test it our model with abstracts from ”SPORT” domain. Table 3.22 shows that our model recognize 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	P	R	F1
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: Outcomes of global model in fine grained run with 10 abstracts from "SPORT" domain

Final experiment that we do with global train model was with "TRANSPORTATION" domain abstracts. 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	P	R	F1
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: Outcomes 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 a 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 waned to reach. Also training such small models takes way more less time, than training a big models.

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

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. 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	P	R	F1
POLITICS	0,9737	0,9610	0,9673
Totals	**0,9737**	**0,9610**	**0,9673**

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

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. 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	P	R	F1
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: Outcome 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.

Table 3.26: TABLE

Entity	P	R	F1
SPORT	1,0000	1,0000	1,0000
Totals	**1,0000**	**1,0000**	**1,0000**

Table 3.27: TABLE

Entity	P	R	F1
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 3.28: TABLE

Entity	**P**	**R**	**F1**
TRANSPORTATION	1,0000	1,0000	1,0000
Totals	**1,0000**	**1,0000**	**1,0000**

Table 3.29: TABLE

Entity	**P**	**R**	**F1**
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 3.30: TABLE

Entity	**P**	**R**	**F1**
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 3.31: TABLE

Entity	**P**	**R**	**F1**
POLITICS	1,0000	0,3906	0,5618
Totals	**1,0000**	**0,3906**	**0,5618**

Table 3.32: TABLE

Entity	**P**	**R**	**F1**
SPORT	1,0000	1,0000	1,0000
Totals	**1,0000**	**1,0000**	**1,0000**

Table 3.33: TABLE

Entity	**P**	**R**	**F1**
SPORT	0,0000	1,0000	0,0000
TRANSPORTATION	1,0000	1,0000	1,0000
Totals	**0,9231**	**1,0000**	**0,9600**

Table 3.34: TABLE

Entity	**P**	**R**	**F1**
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 3.35: TABLE

Entity	**P**	**R**	**F1**
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 3.36: TABLE

Entity	**P**	**R**	**F1**
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 3.37: TABLE

Entity	**P**	**R**	**F1**
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 3.38: TABLE

Entity	**P**	**R**	**F1**
POLITICS	0,9921	0,9766	0,9843
Totals	**0,9655**	**0,3636**	**0,5283**

Table 3.39: TABLE

Entity	**P**	**R**	**F1**
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 3.40: TABLE

Entity	**P**	**R**	**F1**
SPORT	1,0000	1,0000	1,0000
Totals	**1,0000**	**1,0000**	**1,0000**

Table 3.41: TABLE

Entity	**P**	**R**	**F1**
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 3.42: TABLE

Entity	**P**	**R**	**F1**
TRANSPORTATION	1,0000	0,8333	0,9091
Totals	**1,0000**	**0,8333**	**0,9091**

Table 3.43: TABLE

Entity	**P**	**R**	**F1**
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 3.44: TABLE

Entity	**P**	**R**	**F1**
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 3.45: TABLE

Entity	**P**	**R**	**F1**
POLITICS	0,9890	0,3529	0,5202
Totals	**0,9890**	**0,3529**	**0,5202**

Table 3.46: TABLE

Entity	**P**	**R**	**F1**
SPORT	1,0000	1,0000	1,0000
Totals	**1,0000**	**1,0000**	**1,0000**

Table 3.47: TABLE

Entity	**P**	**R**	**F1**
SPORT	0,0000	1,0000	0,0000
TRANSPORTATION	1,0000	0,9846	0,9922
Totals	**0,9697**	**0,9846**	**0,9771**

Table 3.48: TABLE

Entity	**P**	**R**	**F1**
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 3.49: TABLE

Entity	**P**	**R**	**F1**
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 3.50: TABLE

Entity	**P**	**R**	**F1**
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 3.51: TABLE

Entity	**P**	**R**	**F1**
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 3.52: TABLE

Entity	**P**	**R**	**F1**
POLITICS	0,9921	0,9804	0,9862
Totals	**0,9921**	**0,9804**	**0,9862**

Table 3.53: TABLE

Entity	**P**	**R**	**F1**
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 3.54: TABLE

Entity	**P**	**R**	**F1**
SPORT	1,0000	1,0000	1,0000
Totals	**1,0000**	**1,0000**	**1,0000**

Table 3.55: TABLE

Entity	**P**	**R**	**F1**
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 3.56: TABLE

Entity	**P**	**R**	**F1**
TRANSPORTATION	1,0000	0,9846	0,9922
Totals	**1,0000**	**0,9846**	**0,9922**

Table 3.57: TABLE

Entity	**P**	**R**	**F1**
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
SportsTeam	0,0000	1,0000	0,0000
Totals	**1,0000**	**0,9846**	**0,9922**

Table 3.58: TABLE

Entity	**P**	**R**	**F1**
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 3.59: TABLE

Entity	**P**	**R**	**F1**
POLITICS	0,9920	0,3615	0,5299
Totals	**0,9920**	**0,3615**	**0,5299**

Table 3.60: TABLE

Entity	**P**	**R**	**F1**
SPORT	0,9962	0,9888	0,9925
Totals	**0,9962**	**0,9888**	**0,9925**

Table 3.61: TABLE

Entity	**P**	**R**	**F1**
SPORT	0,0000	1,0000	0,0000
TRANSPORTATION	1,0000	0,9735	0,9865
Totals	**0,9821**	**0,9735**	**0,9778**

Table 3.62: TABLE

Entity	**P**	**R**	**F1**
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 3.63: TABLE

Entity	**P**	**R**	**F1**
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 3.64: TABLE

Entity	**P**	**R**	**F1**
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 3.65: TABLE

Entity	**P**	**R**	**F1**
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 3.66: TABLE

Entity	**P**	**R**	**F1**
POLITICS	0,9956	0,9898	0,9927
Totals	**0,9956**	**0,9898**	**0,9927**

Table 3.67: TABLE

Entity	**P**	**R**	**F1**
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 3.68: TABLE

Entity	**P**	**R**	**F1**
SPORT	0,9963	0,9963	0,9963
Totals	**0,9963**	**0,9963**	**0,9963**

Table 3.69: TABLE

Entity	P	R	F1
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 3.70: TABLE

Entity	P	R	F1
TRANSPORTATION	1,0000	0,9912	0,9956
Totals	**1,0000**	**0,9912**	**0,9956**

Table 3.71: TABLE

Entity	P	R	F1
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**

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

Table 3.72: TABLE

Entity	P	R	F1
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 3.73: TABLE

Entity	**P**	**R**	**F1**
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 3.74: TABLE

Entity	**P**	**R**	**F1**
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 3.75: TABLE

Entity	**P**	**R**	**F1**
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 3.76: TABLE

Entity	**P**	**R**	**F1**
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 3.77: TABLE

Entity	**P**	**R**	**F1**
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 3.78: TABLE

Entity	**P**	**R**	**F1**
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 3.79: TABLE

Entity	**P**	**R**	**F1**
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 3.80: TABLE

Entity	**P**	**R**	**F1**
POLITICS	0,9866	0,9479	0,9669
Totals	**0,9866**	**0,9479**	**0,9669**

Table 3.81: TABLE

Entity	**P**	**R**	**F1**
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 3.82: TABLE

Entity	**P**	**R**	**F1**
SPORT	0,9858	0,9676	0,9766
Totals	**0,9858**	**0,8676**	**0,9766**

Table 3.83: TABLE

Entity	**P**	**R**	**F1**
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 3.84: TABLE

Entity	**P**	**R**	**F1**
TRANSPORTATION	0,9954	0,9747	0,9850
Totals	**0,9954**	**0,9747**	**0,9850**

Table 3.85: TABLE

Entity	**P**	**R**	**F1**
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 3.86: TABLE

Entity	**P**	**R**	**F1**
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 3.87: TABLE

Entity	**P**	**R**	**F1**
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 3.88: TABLE

Entity	**P**	**R**	**F1**
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 3.89: TABLE

Entity	**P**	**R**	**F1**
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 3.90: TABLE

Entity	**P**	**R**	**F1**
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 3.91: TABLE

Entity	**P**	**R**	**F1**
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 3.92: TABLE

Entity	**P**	**R**	**F1**
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 3.93: TABLE

Entity	**P**	**R**	**F1**
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 3.94: TABLE

Entity	**P**	**R**	**F1**
POLITICS	0,9808	0,9450	0,9626
Totals	**0,9808**	**0,9450**	**0,9626**

Table 3.95: TABLE

Entity	**P**	**R**	**F1**
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 3.96: TABLE

Entity	**P**	**R**	**F1**
SPORT	0,9856	0,9706	0,9780
Totals	**0,9856**	**0,9706**	**0,9780**

Table 3.97: TABLE

Entity	**P**	**R**	**F1**
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 3.98: TABLE

Entity	**P**	**R**	**F1**
TRANSPORTATION	0,9974	0,9756	0,9864
Totals	**0,9974**	**0,9756**	**0,9864**

Table 3.99: TABLE

Entity	**P**	**R**	**F1**
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**

3.3.4 MIXED

Entity	P	R	F1
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: All 3 Domains Fine Grained Top 300 With All 3 Domains Fine Grained Top 500 Links And All 3 Domains Fine Grained Top 500 Links With Lower PageRank

Entity	P	R	F1
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: All 3 Domains Fine Grained Top 500 Links With All 3 Domains Fine Grained Top 500 Links And All 3 Domains Fine Grained Top 500 Links With Lower Page Rank

Entity	P	R	F1
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: All 3 Domains Fine Grained Top 500 Links With All 3 Domains Fine Grained Top 500 Links With Lower PageRank

Entity	P	R	F1
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: Transportation Fine Grained Top 500 Links Runned With All 3 Domains Fine Grained Top 300 Links And Transportation Fine Grained Top 300 Links

3.3.5 Experiments with lower training abstracts on model, but higher abstracts on test file

```
CRFClassifier tagged 97089 words in 1 documents at 2905,12 words per second.
        Entity P       R       F1      TP      FP      FN
      Aircraft 1,0000  0,0435  0,0833  1       0       22
       Athlete 1,0000  0,0278  0,0541  1       0       35
    Automobile 0,0000  0,0000  0,0000  0       0       3
         Coach 1,0000  0,3333  0,5000  1       0       2
Infrastructure 0,0000  0,0000  0,0000  0       0       22
 PoliticalParty 0,6458  0,3054  0,4147  62      34      141
    Politician 1,0000  0,0727  0,1356  4       0       51
PublicTransitSystem     0,8750  0,1346  0,2333  7       1       45
          Ship 1,0000  0,2000  0,3333  1       0       4
   SpaceShuttle 0,0000  0,0000  0,0000  0       0       6
   SpaceStation 0,0000  0,0000  0,0000  0       0       1
     SportsClub 0,9000  0,1084  0,1935  9       1       74
    SportsEvent 1,0000  0,0233  0,0455  1       0       42
   SportsLeague 0,6667  0,2121  0,3218  14      7       52
  SportsManager 0,0000  0,0000  0,0000  0       0       6
     SportsTeam 1,0000  0,0303  0,0588  1       0       32
          Train 0,0000  0,0000  0,0000  0       0       1
         Totals 0,7034  0,1591  0,2595  102     43      539

C:\Dev\stanford-ner-2017-06-09>java -cp stanford-ner.jar edu.stanford.nlp.ie.crf.CRFClassifier -loadClassifier modelsWit
hOneAnnotation\Top10Links\ner-All3DomainsTop10LinksFineGrained.ser.gz -testFile domainsWithOneAnnotation\Top100Links\All
3DomainsTop100LinksTextWithSameProcessedLinksOnAbstractFineGrainedSpecified.tsv
```

Figure 3.1: All 3 Domains Fine Grained Top 10 Links Runned With All 3 Domains Fine Grained Top 100 Links

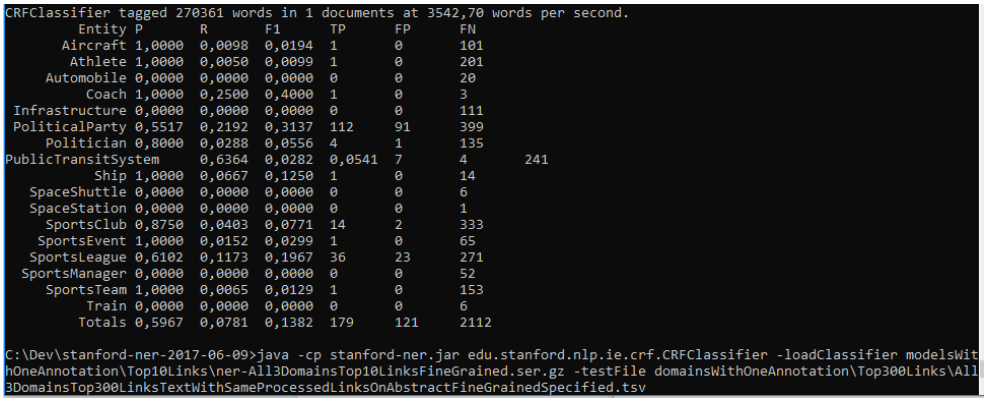

Figure 3.2: All 3 Domains Fine Grained Top 10 Links Runned With All 3 Domains Fine Grained Top 300 Links

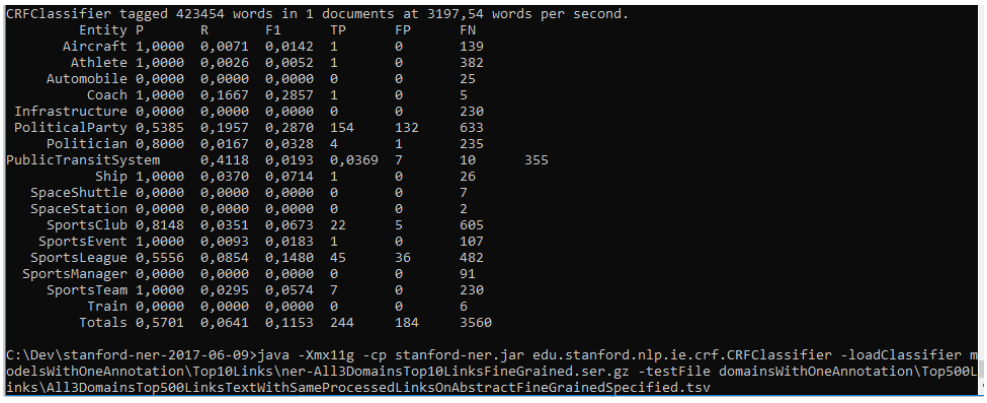

Figure 3.3: All 3 Domains Fine Grained Top 10 Links Runned With All 3 Domains Fine Grained Top 500 Links

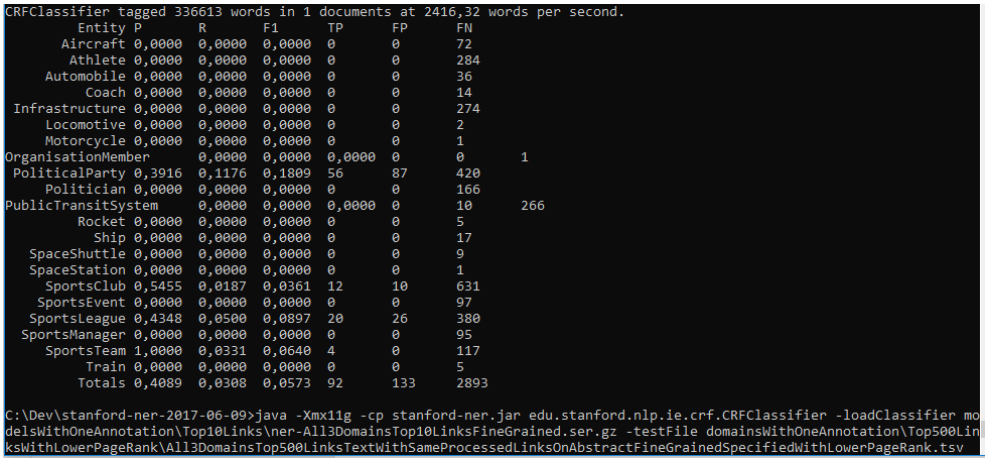

Figure 3.4: All 3 Domains Fine Grained Top 10 Links Runned With All 3 Domains Fine Grained Top 500 Links With Lower PageRank

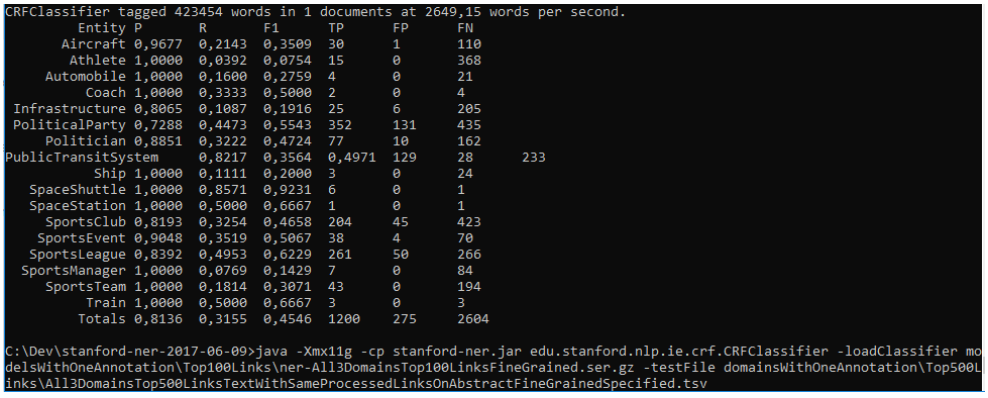

Figure 3.5: All 3 Domains Fine Grained Top 100 Links Runned With All 3 Domains Fine Grained Top 500 Links

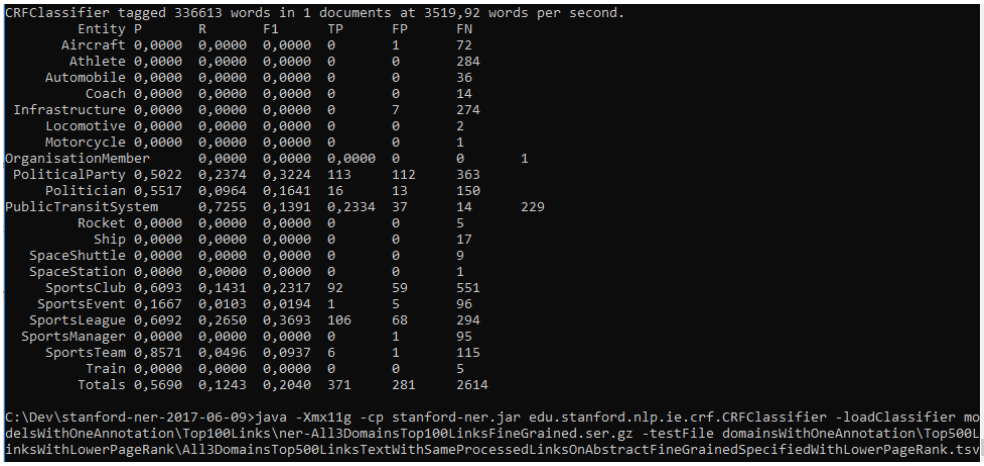

Figure 3.6: All 3 Domains Fine Grained Top 100 Links Runned With All 3 Domains Fine Grained Top 500 Links With Lower PageRank

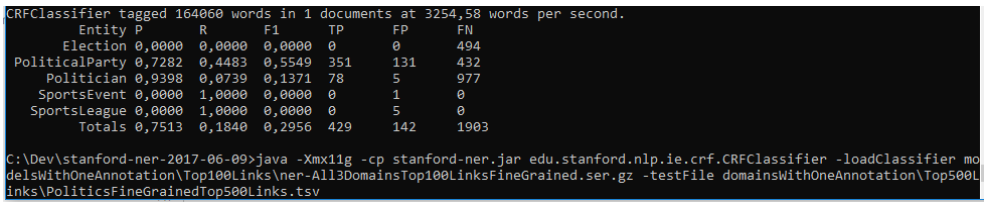

Figure 3.7: All 3 Domains Fine Grained Top 100 Links Runned With Politics Fine Grained Top 500 Links

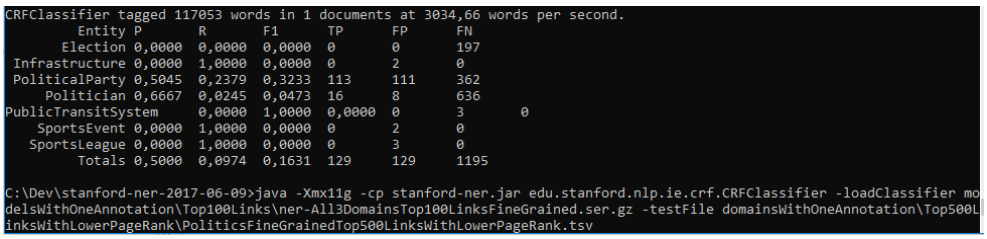

Figure 3.8: All 3 Domains Fine Grained Top 100 Links Runned With Politics Fine Grained Top 500 Links With Lower PageRank

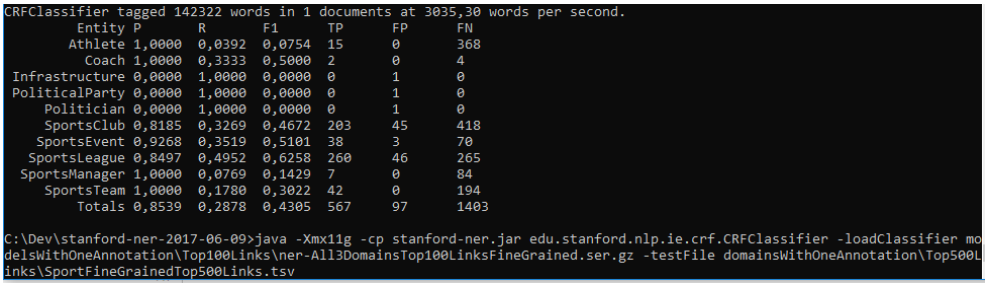

Figure 3.9: All 3 Domains Fine Grained Top 100 Links Runned With Sport Fine Grained Top 500 Links

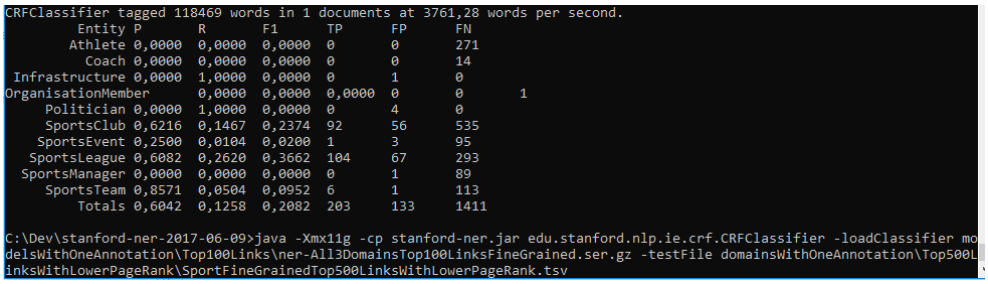

Figure 3.10: All 3 Domains Fine Grained Top 100 Links Runned With Sport Fine Grained Top 500 Links With Lower PageRank

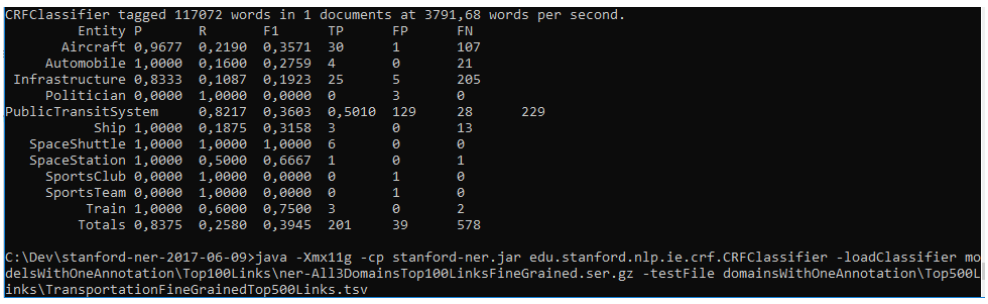

Figure 3.11: All 3 Domains Fine Grained Top 100 Links Runned With Transportation Fine Grained Top 500 Links

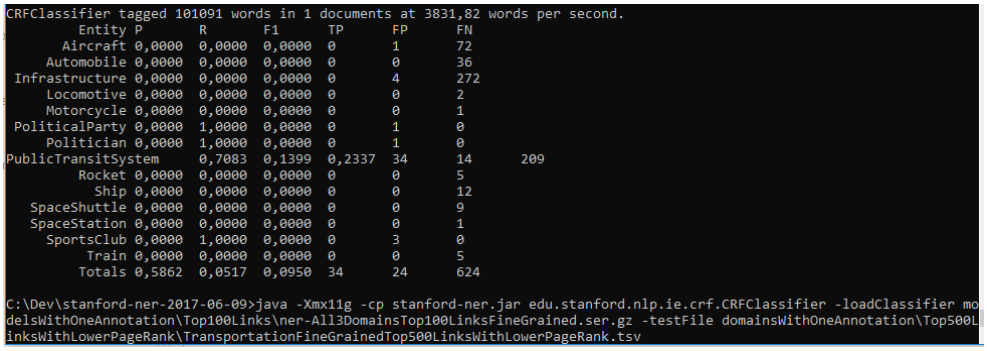

Figure 3.12: All 3 Domains Fine Grained Top 100 Links Runned With Transportation Fine Grained Top 500 Links With Lower PageRank

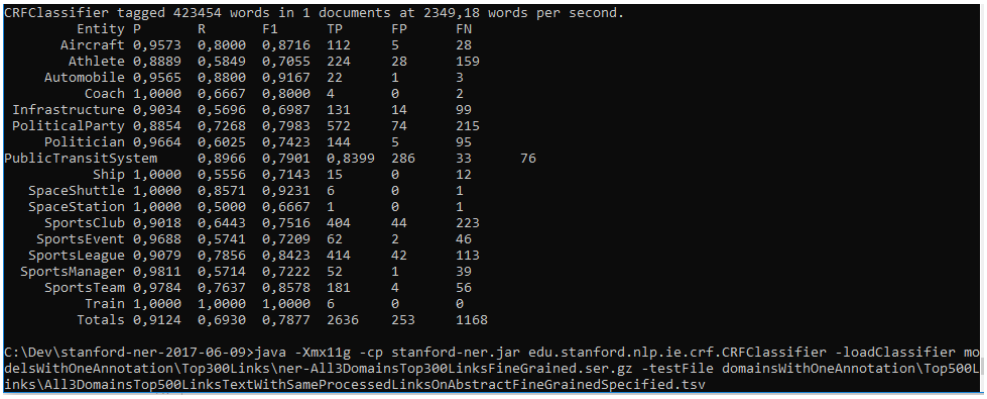

Figure 3.13: All 3 Domains Fine Grained Top 300 Links Runned With All 3 Domains Fine Grained Top 500 Links

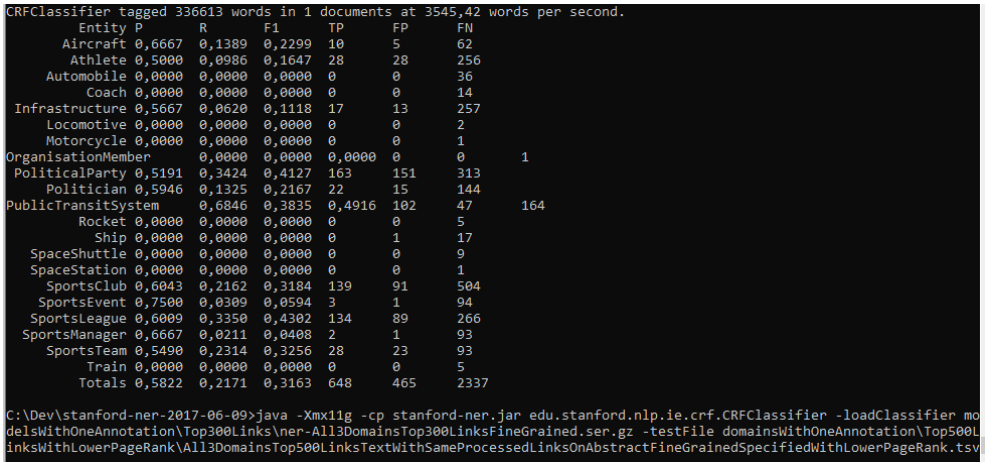

Figure 3.14: All 3 Domains Fine Grained Top 300 Links Runned With All 3 Domains Fine Grained Top 500 Links With Lower PageRank

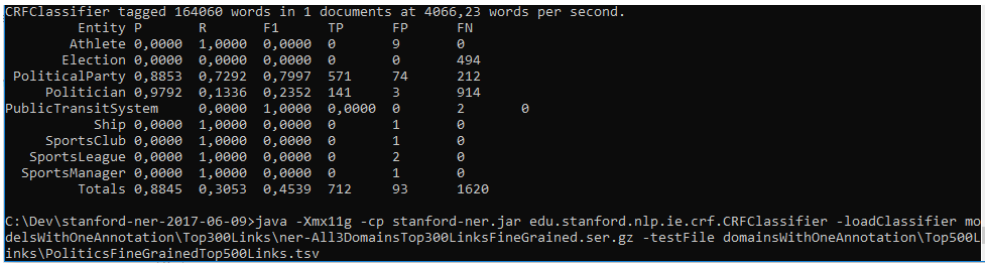

Figure 3.15: All 3 Domains Fine Grained Top 300 Links Runned With Politics Fine Grained Top 500 Links

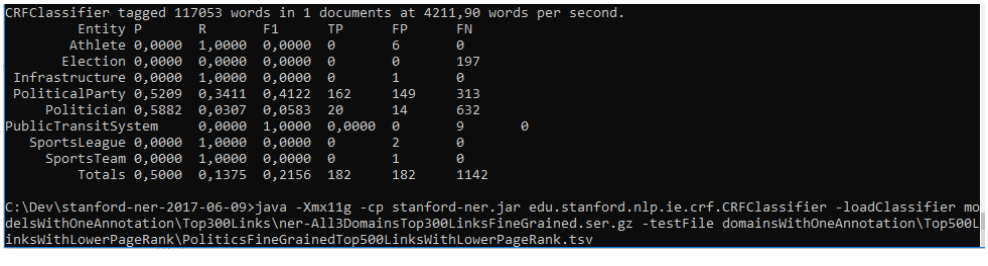

Figure 3.16: All 3 Domains Fine Grained Top 300 Links Runned With Politics Fine Grained Top 500 Links With Lower PageRank

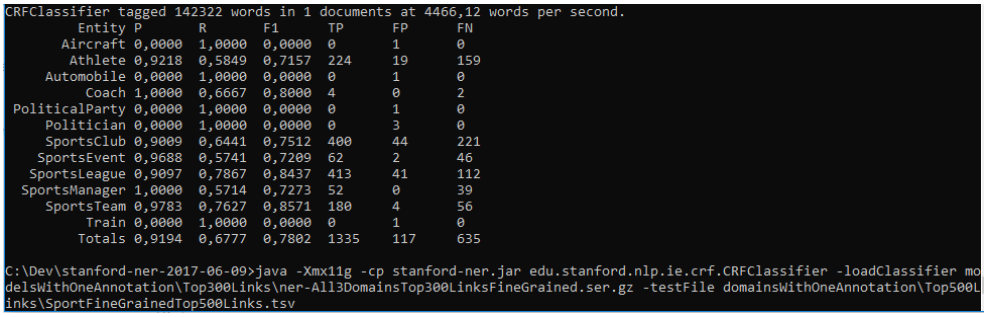

Figure 3.17: All 3 Domains Fine Grained Top 300 Links Runned With Sport Fine Grained Top 500 Links

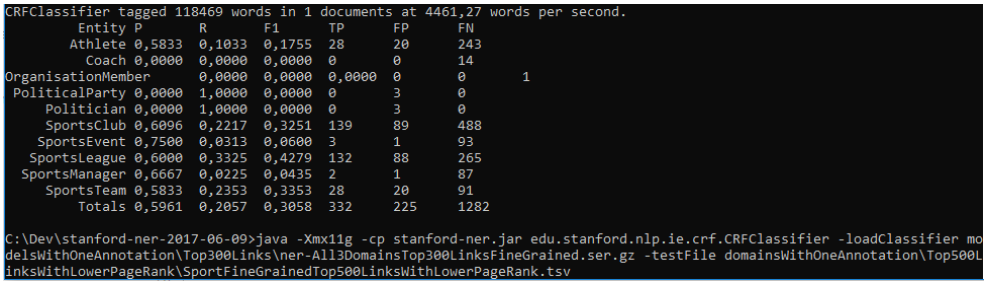

Figure 3.18: All 3 Domains Fine Grained Top 300 Links Runned With Sport Fine Grained Top 500 Links With Lower PageRank

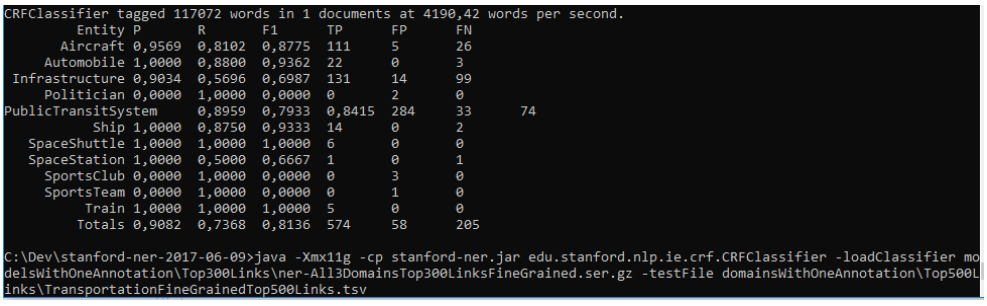

Figure 3.19: All 3 Domains Fine Grained Top 300 Links Runned With Transportation Fine Grained Top 500 Links

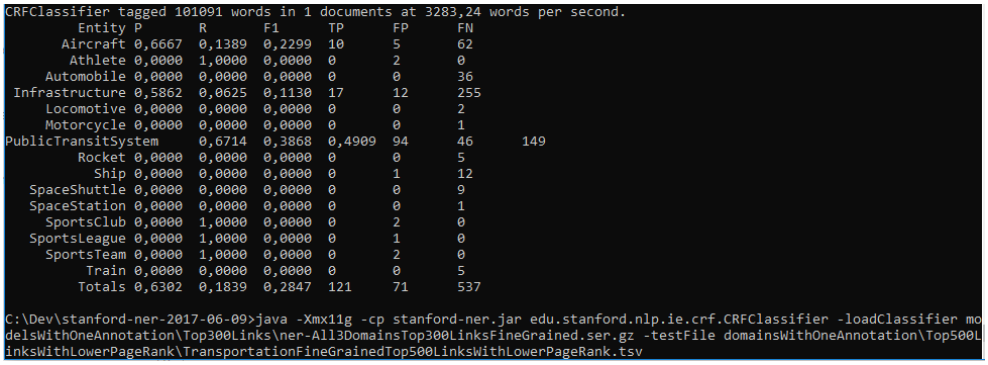

Figure 3.20: All 3 Domains Fine Grained Top 300 Links Runned With Transportation Fine Grained Top 500 Links With Lower PageRank

3.3.6 Experiments with higher training abstracts on model, but lower abstracts on test file

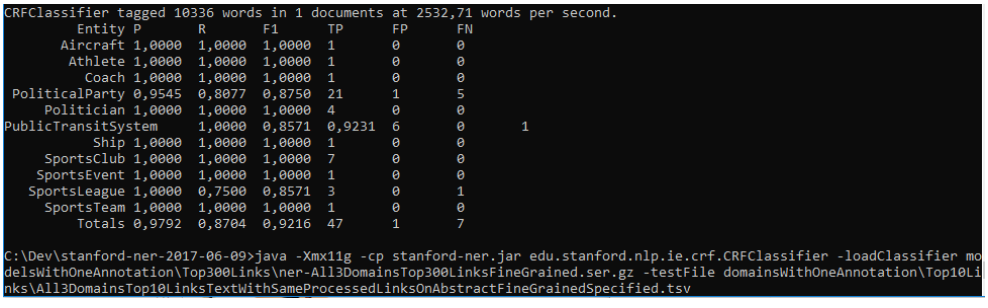

Figure 3.21: All 3 Domains Fine Grained Top 300 Links Runned With All 3 Domains Fine Grained Top 10 Links

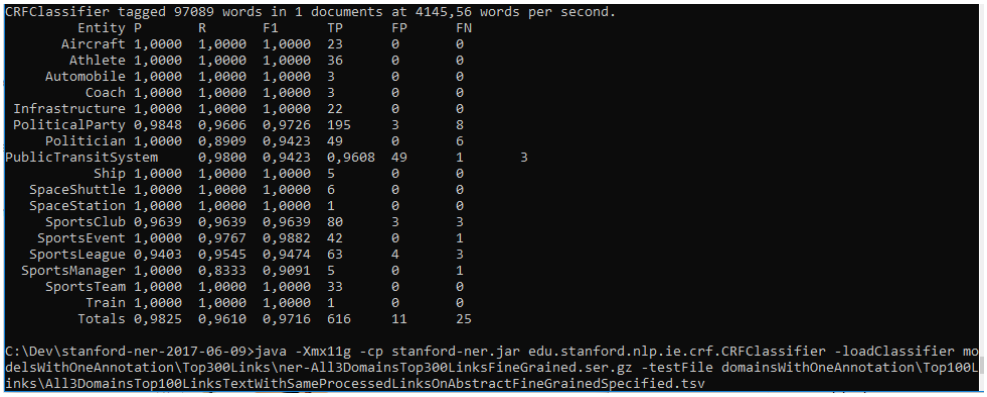

Figure 3.22: All 3 Domains Fine Grained Top 300 Links Runned With All 3 Domains Fine Grained Top 100 Links

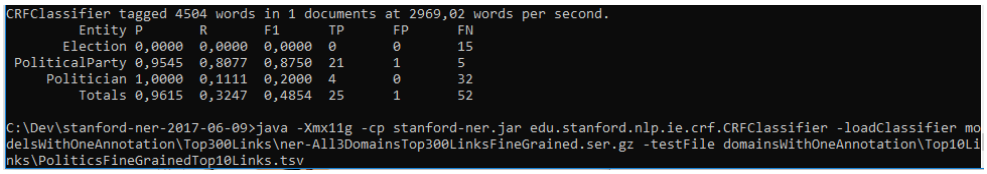

Figure 3.23: All 3 Domains Fine Grained Top 300 Links Runned With Politics Fine Grained Top 10 Links

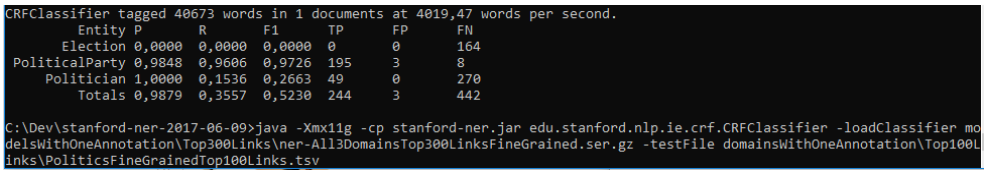

Figure 3.24: All 3 Domains Fine Grained Top 300 Links Runned With Politics Fine Grained Top 100 Links

```
CRFClassifier tagged 2854 words in 1 documents at 2994,75 words per second.
        Entity P       R       F1      TP      FP      FN
       Athlete 1,0000  1,0000  1,0000  1       0       0
         Coach 1,0000  1,0000  1,0000  1       0       0
    SportsClub 1,0000  1,0000  1,0000  7       0       0
   SportsEvent 1,0000  1,0000  1,0000  1       0       0
  SportsLeague 1,0000  0,7500  0,8571  3       0       1
    SportsTeam 1,0000  1,0000  1,0000  1       0       0
        Totals 1,0000  0,9333  0,9655  14      0       1

C:\Dev\stanford-ner-2017-06-09>java -Xmx11g -cp stanford-ner.jar edu.stanford.nlp.ie.crf.CRFClassifier -loadClassifier mo
delsWithOneAnnotation\Top300Links\ner-All3DomainsTop300LinksFineGrained.ser.gz -testFile domainsWithOneAnnotation\Top10Li
nks\SportFineGrainedTop10Links.tsv
```

Figure 3.25: All 3 Domains Fine Grained Top 300 Links Runned With Sport Fine Grained Top 10 Links

```
CRFClassifier tagged 31872 words in 1 documents at 3691,88 words per second.
        Entity P       R       F1      TP      FP      FN
       Athlete 1,0000  1,0000  1,0000  36      0       0
         Coach 1,0000  1,0000  1,0000  3       0       0
    SportsClub 0,9634  0,9634  0,9634  79      3       3
   SportsEvent 1,0000  0,9767  0,9882  42      0       1
  SportsLeague 0,9403  0,9545  0,9474  63      4       3
 SportsManager 1,0000  0,8333  0,9091  5       0       1
    SportsTeam 1,0000  1,0000  1,0000  32      0       0
        Totals 0,9738  0,9701  0,9720  260     7       8

C:\Dev\stanford-ner-2017-06-09>java -Xmx11g -cp stanford-ner.jar edu.stanford.nlp.ie.crf.CRFClassifier -loadClassifier mo
delsWithOneAnnotation\Top300Links\ner-All3DomainsTop300LinksFineGrained.ser.gz -testFile domainsWithOneAnnotation\Top100Li
nks\SportFineGrainedTop100Links.tsv
```

Figure 3.26: All 3 Domains Fine Grained Top 300 Links Runned With Sport Fine Grained Top 100 Links

```
CRFClassifier tagged 2978 words in 1 documents at 2863,46 words per second.
        Entity P       R       F1      TP      FP      FN
      Aircraft 1,0000  1,0000  1,0000  1       0       0
PublicTransitSystem     1,0000  0,8571  0,9231  6       0       1
          Ship 1,0000  1,0000  1,0000  1       0       0
        Totals 1,0000  0,8889  0,9412  8       0       1

C:\Dev\stanford-ner-2017-06-09>java -Xmx11g -cp stanford-ner.jar edu.stanford.nlp.ie.crf.CRFClassifier -loadClassifier mo
delsWithOneAnnotation\Top300Links\ner-All3DomainsTop300LinksFineGrained.ser.gz -testFile domainsWithOneAnnotation\Top10Li
nks\TransportationFineGrainedTop10Links.tsv
```

Figure 3.27: All 3 Domains Fine Grained Top 300 Links Runned With Transportation Fine Grained Top 10 Links

```
CRFClassifier tagged 24544 words in 1 documents at 4180,55 words per second.
        Entity P       R       F1      TP      FP      FN
      Aircraft 1,0000  1,0000  1,0000  23      0       0
    Automobile 1,0000  1,0000  1,0000  3       0       0
 Infrastructure 1,0000  1,0000  1,0000  22      0       0
PublicTransitSystem     0,9800  0,9423  0,9608  49      1       3
          Ship 1,0000  1,0000  1,0000  5       0       0
   SpaceShuttle 1,0000  1,0000  1,0000  6       0       0
   SpaceStation 1,0000  1,0000  1,0000  1       0       0
     SportsClub 0,0000  1,0000  0,0000  0       1       0
     SportsTeam 0,0000  1,0000  0,0000  0       1       0
          Train 1,0000  1,0000  1,0000  1       0       0
         Totals 0,9735  0,9735  0,9735  110     3       3

C:\Dev\stanford-ner-2017-06-09>java -Xmx11g -cp stanford-ner.jar edu.stanford.nlp.ie.crf.CRFClassifier -loadClassifier mo
delsWithOneAnnotation\Top300Links\ner-All3DomainsTop300LinksFineGrained.ser.gz -testFile domainsWithOneAnnotation\Top100L
inks\TransportationFineGrainedTop100Links.tsv
```

Figure 3.28: All 3 Domains Fine Grained Top 300 Links Runned With Transportation Fine Grained Top 100 Links

```
CRFClassifier tagged 10336 words in 1 documents at 3459,17 words per second.
        Entity P       R       F1      TP      FP      FN
      Aircraft 1,0000  1,0000  1,0000  1       0       0
       Athlete 1,0000  1,0000  1,0000  1       0       0
         Coach 1,0000  1,0000  1,0000  1       0       0
 PoliticalParty 0,9545  0,8077  0,8750  21      1       5
     Politician 1,0000  1,0000  1,0000  4       0       0
PublicTransitSystem     0,8333  0,7143  0,7692  5       1       2
          Ship 1,0000  1,0000  1,0000  1       0       0
    SportsClub 0,8750  1,0000  0,9333  7       1       0
   SportsEvent 1,0000  1,0000  1,0000  1       0       0
  SportsLeague 1,0000  1,0000  1,0000  4       0       0
    SportsTeam 1,0000  1,0000  1,0000  1       0       0
        Totals 0,9400  0,8704  0,9038  47      3       7

C:\Dev\stanford-ner-2017-06-09>java -Xmx11g -cp stanford-ner.jar edu.stanford.nlp.ie.crf.CRFClassifier -loadClassifier mo
delsWithOneAnnotation\Top500Links\ner-All3DomainsTop500LinksFineGrained.ser.gz -testFile domainsWithOneAnnotation\Top10Li
nks\All3DomainsTop10LinksTextWithSameProcessedLinksOnAbstractFineGrainedSpecified.tsv
```

Figure 3.29: All 3 Domains Fine Grained Top 500 Links Runned With All 3 Domains Fine Grained Top 10 Links

```
CRFClassifier tagged 97089 words in 1 documents at 3675,94 words per second.
        Entity P       R       F1      TP      FP      FN
      Aircraft 1,0000  1,0000  1,0000  23      0       0
       Athlete 1,0000  1,0000  1,0000  36      0       0
    Automobile 1,0000  1,0000  1,0000  3       0       0
         Coach 1,0000  1,0000  1,0000  3       0       0
 Infrastructure 1,0000  0,9545  0,9767  21      0       1
 PoliticalParty 0,9799  0,9606  0,9701  195     4       8
     Politician 1,0000  1,0000  1,0000  55      0       0
PublicTransitSystem     0,9796  0,9231  0,9505  48      1       4
          Ship 1,0000  1,0000  1,0000  5       0       0
   SpaceShuttle 1,0000  1,0000  1,0000  6       0       0
   SpaceStation 1,0000  1,0000  1,0000  1       0       0
     SportsClub 0,9529  0,9759  0,9643  81      4       2
    SportsEvent 1,0000  0,8837  0,9383  38      0       5
   SportsLeague 0,9275  0,9697  0,9481  64      5       2
  SportsManager 1,0000  1,0000  1,0000  6       0       0
     SportsTeam 1,0000  1,0000  1,0000  33      0       0
          Train 1,0000  1,0000  1,0000  1       0       0
         Totals 0,9779  0,9657  0,9717  619     14      22

C:\Dev\stanford-ner-2017-06-09>java -Xmx11g -cp stanford-ner.jar edu.stanford.nlp.ie.crf.CRFClassifier -loadClassifier mo
delsWithOneAnnotation\Top500Links\ner-All3DomainsTop500LinksFineGrained.ser.gz -testFile domainsWithOneAnnotation\Top100L
inks\All3DomainsTop100LinksTextWithSameProcessedLinksOnAbstractFineGrainedSpecified.tsv
```

Figure 3.30: All 3 Domains Fine Grained Top 500 Links Runned With All 3 Domains Fine Grained Top 100 Links

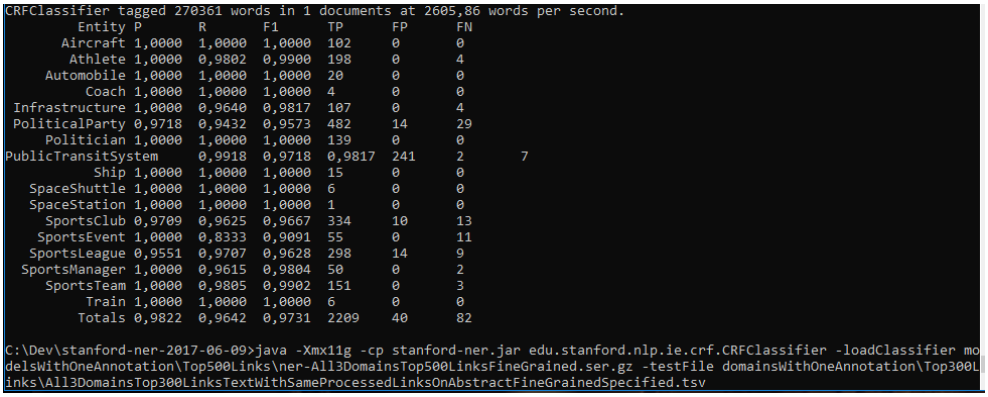

Figure 3.31: All 3 Domains Fine Grained Top 500 Links Runned With All 3 Domains Fine Grained Top 300 Links

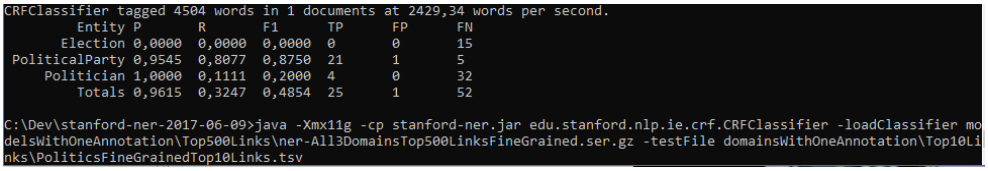

Figure 3.32: All 3 Domains Fine Grained Top 500 Links Runned With Politics Fine Grained Top 10 Links

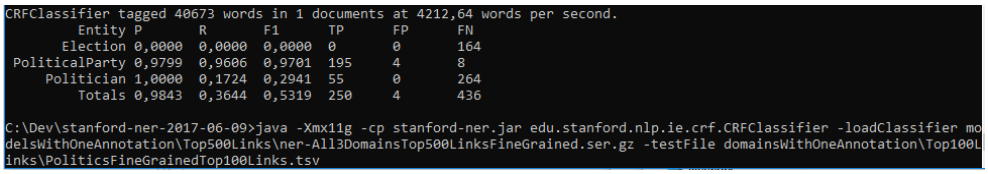

Figure 3.33: All 3 Domains Fine Grained Top 500 Links Runned With Politics Fine Grained Top 100 Links

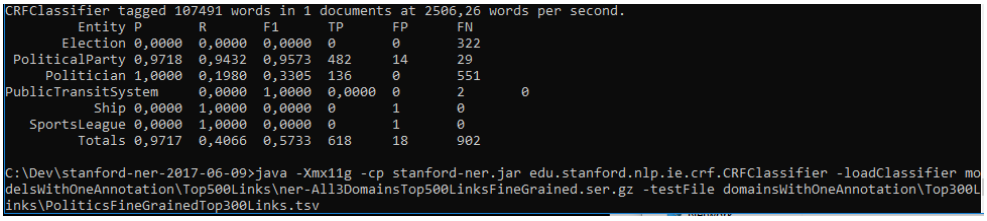

Figure 3.34: All 3 Domains Fine Grained Top 500 Links Runned With Politics Fine Grained Top 300 Links

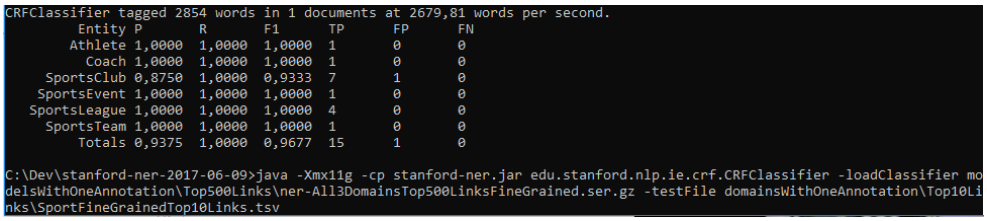

Figure 3.35: All 3 Domains Fine Grained Top 500 Links Runned With Sport Fine Grained Top 10 Links

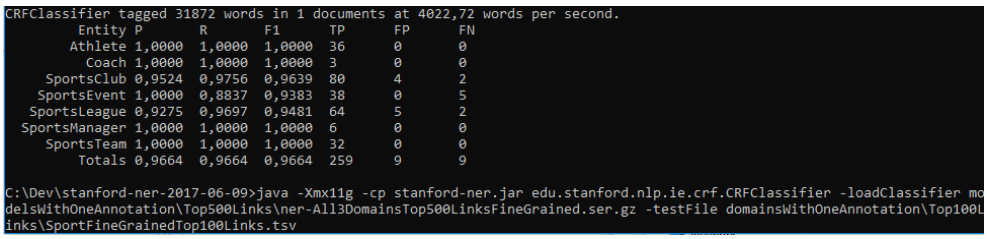

Figure 3.36: All 3 Domains Fine Grained Top 500 Links Runned With Sport Fine Grained Top 100 Links

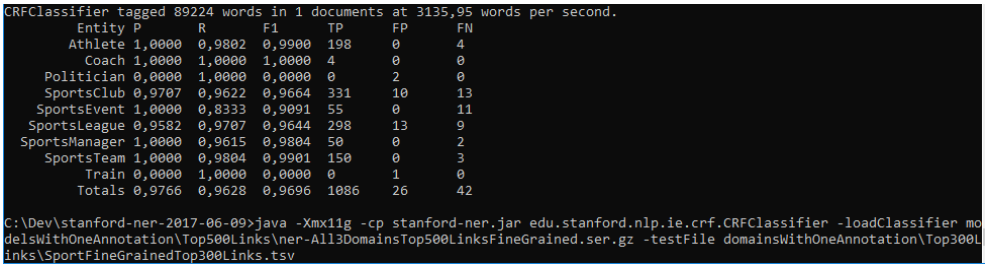

Figure 3.37: All 3 Domains Fine Grained Top 500 Links Runned With Sport Fine Grained Top 300 Links

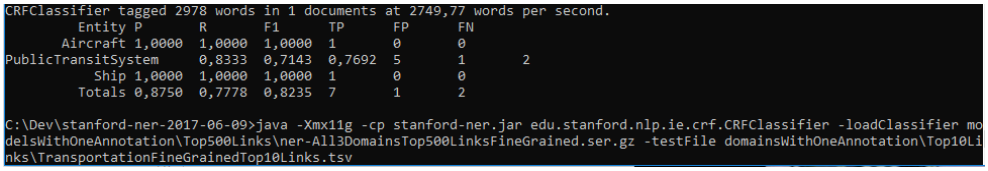

Figure 3.38: All 3 Domains Fine Grained Top 500 Links Runned With Transportation Fine Grained Top 10 Links

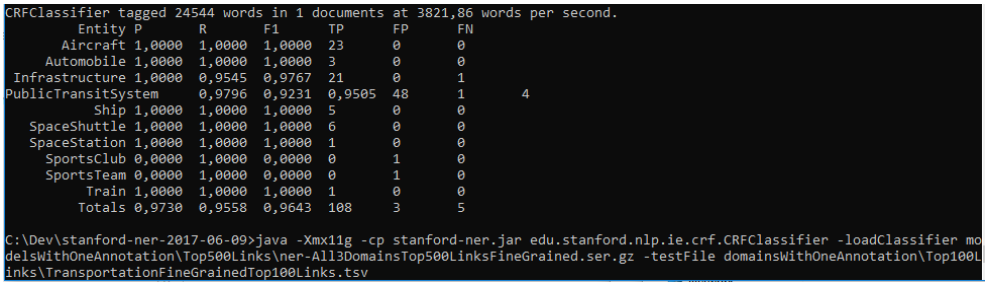

Figure 3.39: All 3 Domains Fine Grained Top 500 Links Runned With Transportation Fine Grained Top 100 Links

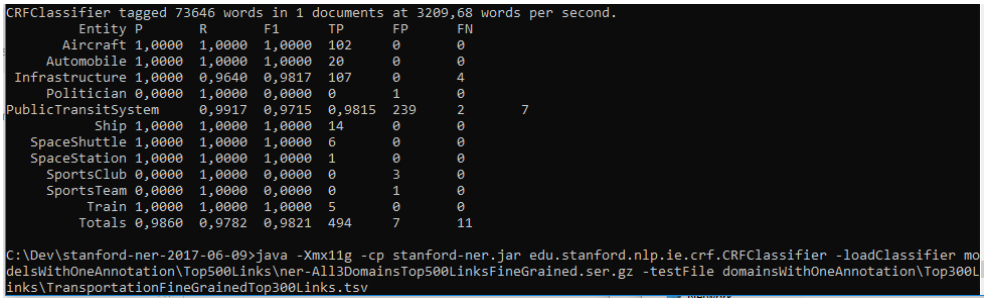

Figure 3.40: All 3 Domains Fine Grained Top 500 Links Runned With Transportation Fine Grained Top 300 Links

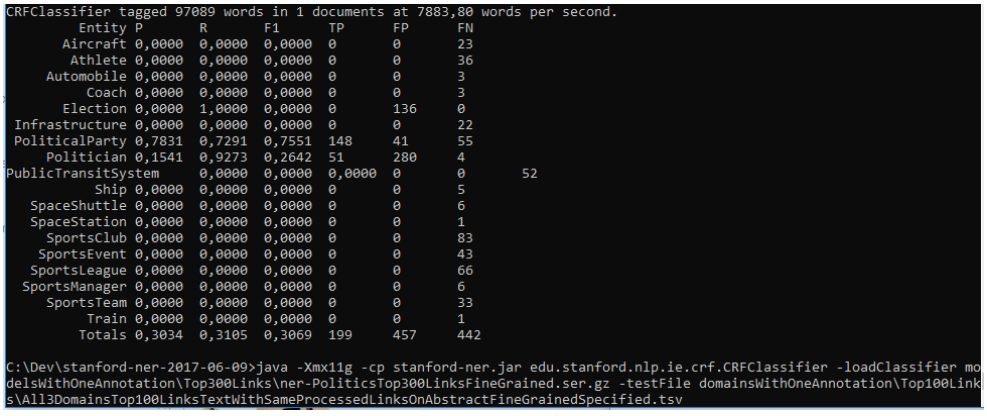

Figure 3.41: Politics Fine Grained Top 300 Links Runned With All 3 Domains Fine Grained Top 100 Links

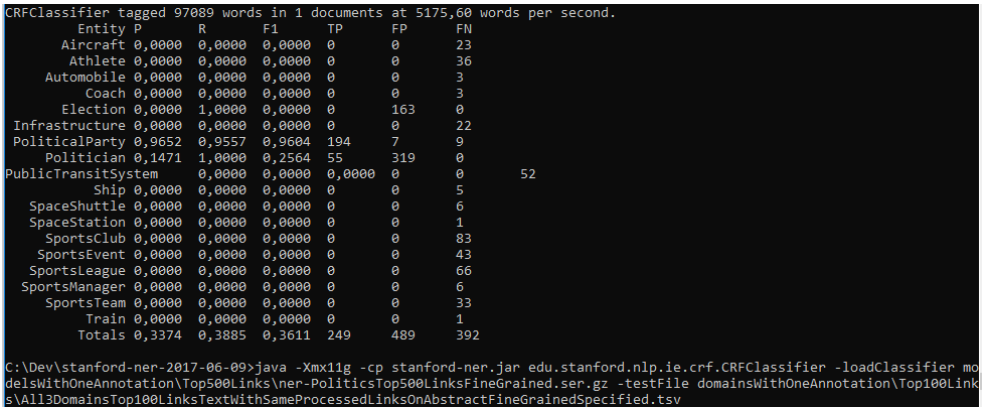

Figure 3.42: Politics Fine Grained Top 500 Links Runned With All 3 Domains Fine Grained Top 100 Links

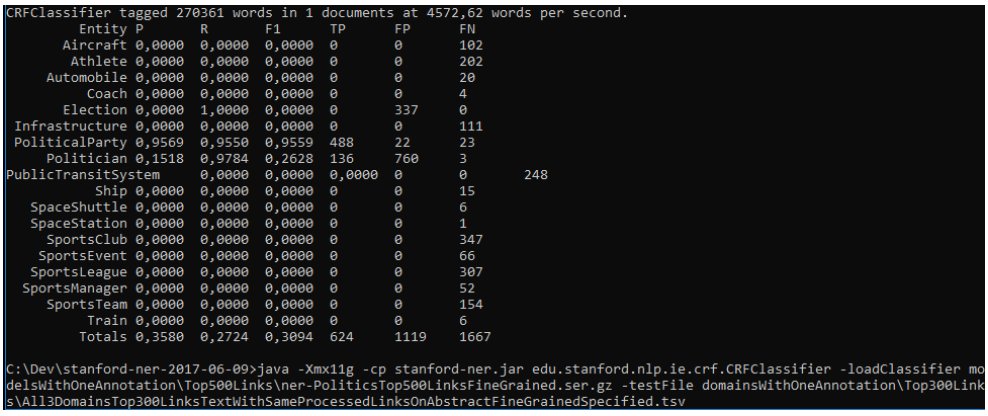

Figure 3.43: Politics Fine Grained Top 500 Links Runned With All 3 Domains Fine Grained Top 300 Links

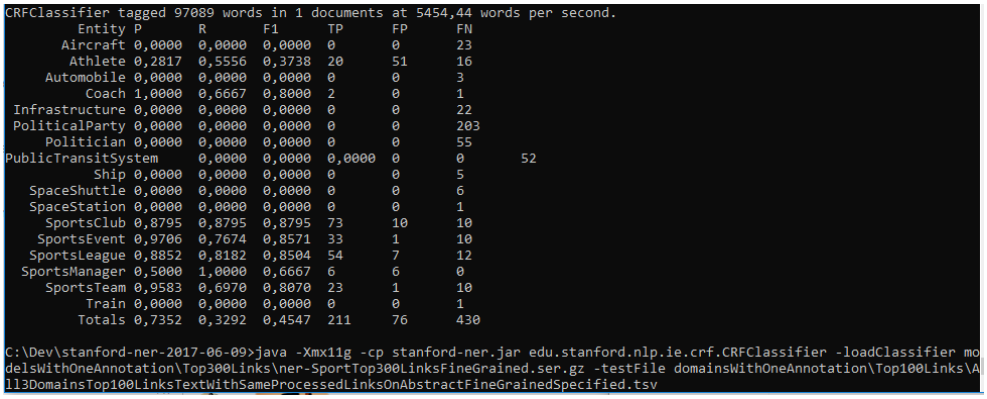

Figure 3.44: Sport Fine Grained Top 300 Links Runned With All 3 Domains Fine Grained Top 100 Links

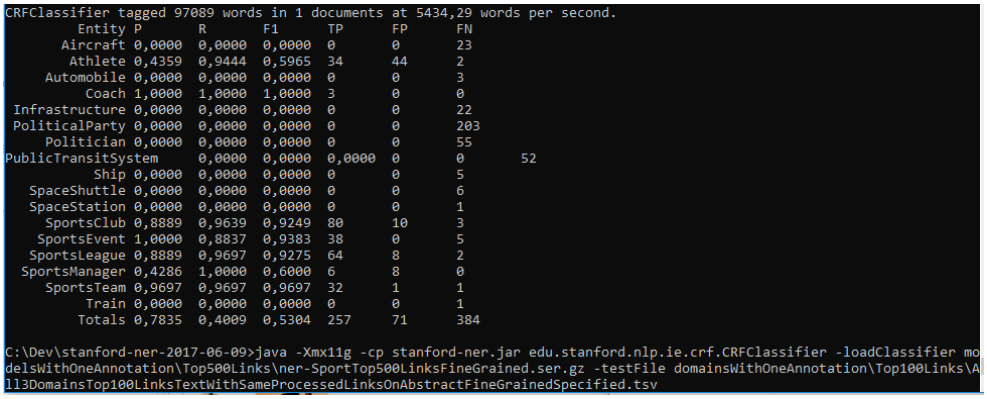

Figure 3.45: Sport Fine Grained Top 500 Links Runned With All 3 Domains Fine Grained Top 100 Links

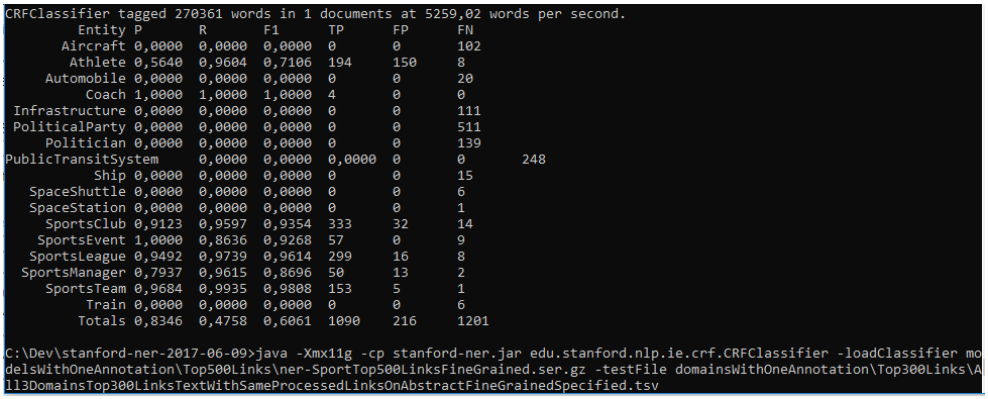

Figure 3.46: Sport Fine Grained Top 500 Links Runned With All 3 Domains Fine Grained Top 300 Links

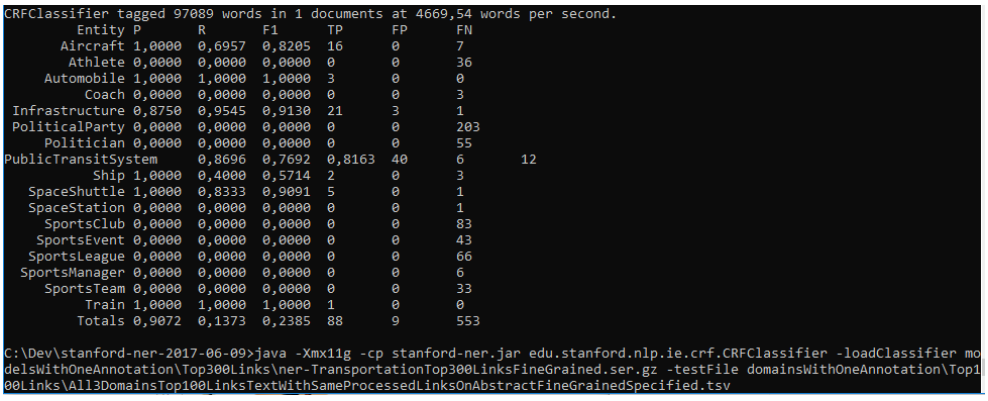

Figure 3.47: Transportation Fine Grained Top 300 Links Runned With All 3 Domains Fine Grained Top 100 Links

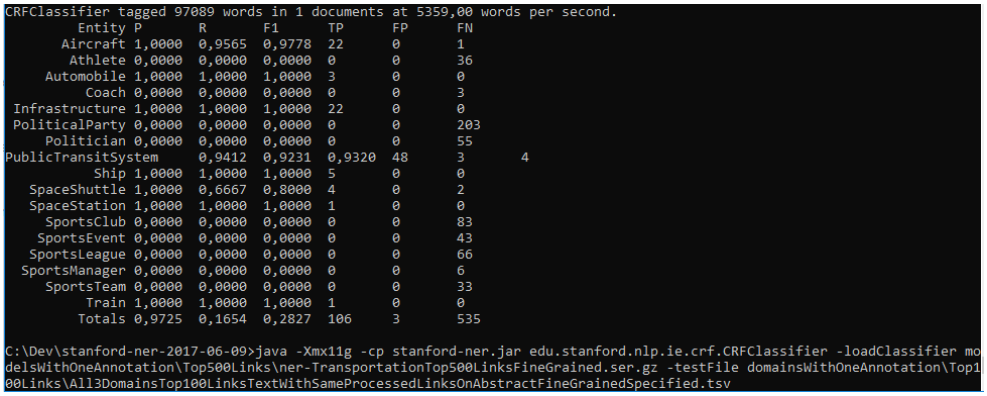

Figure 3.48: Transportation Fine Grained Top 500 Links Runned With All 3 Domains Fine Grained Top 100 Links

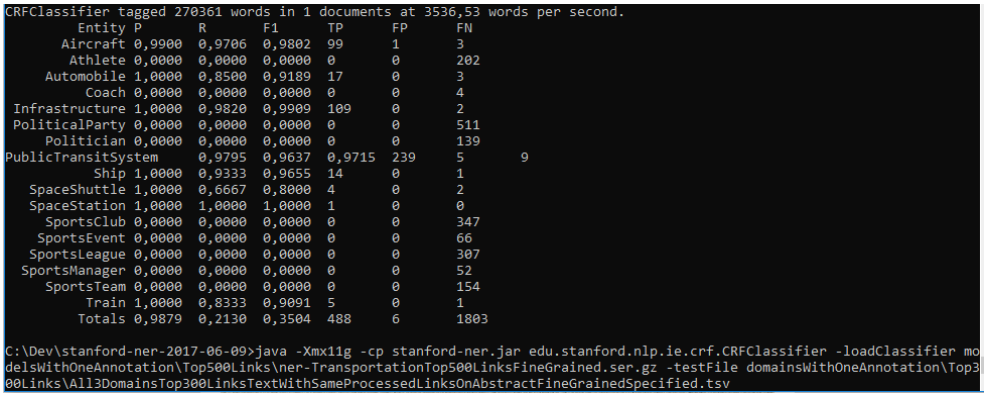

Figure 3.49: Transportation Fine Grained Top 500 Links Runned With All 3 Domains Fine Grained Top 300 Links

3.4 Summary of results

3.4.1 Graphs

Conclusion

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

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

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

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

[18] 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

[19] 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

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

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

Appendix A

Retrieved types

A.1 POLITICS types

Parliament, Election,PoliticalParty, GeopoliticalOrganisation, Politician, Ambassador, Chancellor, Congressman, Deputy, Governor, Lieutenant, Mayor, MemberOfParliament, Minister, President, PrimeMinister, Senator, VicePresident, VicePrimeMinister, PoliticianSpouse, PersonFunction, PoliticalFunction, Profession, TopicalConcept and PoliticalConcept.

A.2 SPORT 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, BoxingLeague, 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.3 TRANSPORTATION types

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

Types: Automobile, automobilePlatform, bodyStyle, enginePower, engineType, powerType, transmission and AutomobileEngine are grouped at Automobile type.

Types: Locomotive, boiler, boilerPressure and cylinderCount are grouped at Locomotive type.

Types: MilitaryVehicle, Motorcycle and SpaceStation are not grouped, this are leaved as it is.

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

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

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

Appendix B

Stanford NER properties file

Here is the example of one of the properties file that we use for creating model with all used flags:

```
# 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
```

Appendix C

Contents of CD