
Pós-Graduação em Ciência da Computação

JOHNY MOREIRA DA SILVA

# Extracting Structured Information from Text to Augment Knowledge Bases

Universidade Federal de Pernambuco
posgraduacao@cin.ufpe.br
http://cin.ufpe.br/~posgraduacao

Recife
2019

JOHNY MOREIRA DA SILVA

# Extracting Structured Information from Text to Augment Knowledge Bases

Trabalho apresentado ao Programa de Pós-graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de Mestre em Ciência da Computação.

**Área de Concentração**: Banco de Dados

**Orientador**: Luciano de Andrade Barbosa

Recife
2019

Catalogação na fonte
Bibliotecária Monick Raquel Silvestre da S. Portes, CRB4-1217

S586e Silva, Johny Moreira da
*Extracting structured information from text to augment knowledge bases* / Johny Moreira da Silva. – 2019.
93 f.: il., fig., tab.

Orientador: Luciano de Andrade Barbosa.
Dissertação (Mestrado) – Universidade Federal de Pernambuco. CIn, Ciência da Computação, Recife, 2019.
Inclui referências e apêndices.

1. Banco de dados. 2. Processamento de linguagem natural. I. Barbosa, Luciano de Andrade (orientador). II. Título.

025.04 CDD (23. ed.) UFPE- MEI 2019-080

**Johny Moreira da Silva**

**“Extracting Structured Information from Text to Augment Knowledge Bases”**

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Pernambuco, como requisito parcial para a obtenção do título de Mestre em Ciência da Computação.

Aprovado em: 25/02/2019.

**BANCA EXAMINADORA**

_____

Prof. Dr. Luciano de Andrade Barbosa (Orientador)
Universidade Federal de Pernambuco

_____

Prof. Dr. Paulo Salgado Gomes de Mattos Neto (Examinador Interno)
Universidade Federal de Pernambuco

_____

Prof. Dr. Hendrik Teixeira Macedo (Examinador Externo)
Universidade Federal de Sergipe

To my mother, Maria Delvani. For her strength, love and faith. The kindest person I've ever known. With love.

## ACKNOWLEDGEMENTS

Firstly, I would like to express my sincere gratitude to my advisor Prof. Luciano de Andrade Barbosa for the continuous support of my Masters study, for his patience, motivation, presence, and immense knowledge. His guidance helped me in all the time of research and writing of this Master thesis. Thank you for sharing your knowledge and enthusiasm.

Besides my advisor, I would like to thank my professors at PPGCC (Postgraduate Program in Science Computing) and former professors, who have guided me through this journey in the searching for knowledge. Sharing knowledge is one of the noblest and admirable attitudes I'm aware of, and you exercised it adroitly. You'll always be remembered.

My sincere gratitude to the members of my examination board: Prof. Paulo Salgado Gomes de Mattos Neto, and Prof. Hendrik Teixeira Macedo, for their insightful comments, feedbacks and contributions to improving this work.

Very special gratitude goes out to FACEPE (Fundação de Amparo a Ciência e Tecnologia do Estado de Pernambuco) for providing the funding for the work.

Thank you to Chaina Oliveira, Raphael Dourado, Everaldo Neto, and Junior Sobreira for helping me with the construction of the datasets used to validate this work. You Rock!

I would like to express my very profound gratitude to my family for providing me with unfailing support and continuous encouragement throughout my years of study and my life in general. Thanks for understanding my absence in important moments for the last few years. This accomplishment would not have been possible without you.

Last but not least, I thank all my friends, which I will not personally mention to not fall in the mistake of missing someone. Thanks for the chatting, the drinkings, the trips, the encouragement, and enthusiasm. The path without you would be really boring.

Thank you.

## ABSTRACT

Knowledge graphs (or knowledge bases) allow data organization and exploration, making easier the semantic understanding and use of data by machines. Traditional strategies for knowledge base construction have mostly relied on manual effort, or have been automatically extracted from structured and semi-structured data. Considering the large amount of unstructured information on the Web, new approaches on knowledge bases construction and maintenance are trying to leverage this information to improve the quality and coverage of knowledge graphs. In this work, focusing in the completeness problem of existing knowledge bases, we are interested in extracting from unstructured text missing attributes of entities in knowledge bases. For this study, in particular, we use the infoboxes of entities in Wikipedia articles as instances of the knowledge graph and their respective text as source of unstructured data. More specifically, given Wikipedia articles of entities in a particular domain, the structured information of the entity's attributes in the infobox is used by a distant supervision strategy to identify sentences that mention those attributes in the text. These sentences are provided as labels to train a sequence-based neural network (Bidirectional Long Short-Term Memory or Convolutional Neural Network), which then performs the extraction of the attributes on unseen articles. We have compared our strategy with two traditional approaches for this problem, Kylin and iPopulator. Our distant supervision model have presented a considerable amount of positive and negative training examples, obtaining representative training examples when compared with the other two traditional systems. Also, our pipeline extraction have shown better performance filling the proposed schema. Overall, the extraction pipeline proposed in this work outperforms the baseline models with an average increase of 0.29 points in F-Score, showing significant difference in performance. In this work we have proposed a modification of the Distant Supervision paradigm for automatic labeling of training examples and an extraction pipeline for filling out a given schema with better performance than the analyzed baseline systems.



**Keywords**: Natural Language Processing. Named Entity Recognition. Wikipedia. DBpedia. Deep Learning. Distant Supervision.

## RESUMO

Grafos de Conhecimento (ou Bases de Conhecimento) permitem a organização e exploração de dados, tornando mais fácil o seu entendimento semântico e utilização por máquinas. Estratégias tradicionais para construção de bases de conhecimento tem dependido na maior parte das vezes de esforço manual, ou tem utilizado extração automática de fontes de dados estruturadas e semi-estruturadas. Considerando a grande quantidade de informação não estruturada na Web, novas abordagens para construção e manutenção de bases de conhecimento tem tentado alavancar o uso dessa fonte como forma de melhorar a qualidade e a cobertura dos grafos de conhecimento. Este trabalho está voltado para o problema de completude de bases de conhecimento, nós estamos interessados em extrair de textos não estruturados os atributos faltosos de entidades. Para este estudo em particular, nós fazemos uso de Infoboxes de entidades de artigos da Wikipédia como instâncias do grafo de conhecimento, e os textos desses artigos são utilizados como fonte de dados não estruturados. Mais especificamente, dados artigos de entidades da Wikipédia de um determinado domínio, a informação estruturada dos atributos de Infobox da entidade são usados por uma estratégia de supervisão distante, de forma a identificar sentenças que mencionam esses atributos. Essas sentenças são rotuladas e utilizadas para treino de uma rede neural baseada em sequência (Rede Bidirecional de Memória de Curto-Longo Prazo ou Rede Neural Convolucional), que realizam a extração de atributos em novos artigos. Nós comparamos nossa estratégia com duas abordagens tradicionais para o mesmo problema, Kylin e iPopulator. Nosso modelo de supervisão distante apresentou uma quantidade considerável de exemplos de treinamento positivos e negativos quando comparado com os outros dois sistemas tradicionais. Nosso esquema de extração também apresentou melhor performance no preenchimento do esquema de dados proposto. No geral, nosso sistema de extração superou os modelos de base com um aumento médio de 0.29 pontos no F-Score, mostrando diferença significativa de performance. Neste trabalho foi proposto uma modificação do paradigma de supervisão distante para rotulagem automática de exemplos de treinamento, e um esquema de extração para preenchimento de um dado esquema de dados com performance superior aos sistemas de base analisados.



**Palavras-chave**: Processamento de Linguagem Natural. Reconhecimento de Entidades Nomeadas. Wikipedia. DBpedia. Aprendizagem Profunda. Supervisão Distante.

## LIST OF FIGURES

## LIST OF TABLES

## LIST OF ABBREVIATIONS AND ACRONYMS

| | |
|---|---|
| biRNN | Bidirectional Recurrent Neural Networks |
| BLSTM | Bidirectional Long Short-Term Memory |
| BoW | Bag-of-Words |
| CNN | Convolutional Neural Networks |
| CRF | Conditional Random Fields |
| CSS | Cascading Style Sheets |
| DL | Deep Learning |
| DS | Distant Supervision |
| FN | False Negative |
| FP | False Positive |
| HMM | Hidden Markov Model |
| HTML | Hypertext Markup Language |
| IE | Information Extraction |
| KB | Knowledge Base |
| KBC | Knowledge Base Construction |
| KG | Knowledge Graph |
| L-BFGS | Limited-memory Broyden–Fletcher–Goldfarb–Shanno |
| LOD | Linked Open Data |
| LSTM | Long Short-Term Memory |
| ML | Machine Learning |
| NER | Named Entity Recognition |
| NIF | NLP Interchange Format |
| NLP | Natural Language Processing |
| OpenIE | Open Information Extraction |
| POS | Part of Speech |
| RDF | Resource Description Framework |
| RE | Relation Extraction |
| RNN | Recurrent Neural Networks |

| | |
|---|---|
| SKOS | Simple Knowledge Organization System |
| SVM | Support Vector Machine |
| TF-IDF | Term Frequency-Inverse Document Frequency |
| TN | True Negative |
| TP | True Positive |
| URI | Uniform Resource Identifier |
| URL | Uniform Resource Locator |
| W3C | World Wide Web Consortium |
| WWW | World Wide Web |
| XML | Extensible Markup Language |

## LIST OF SYMBOLS

| | |
|---|---|
| $\gamma$ | Greek letter gamma |
| $\Gamma$ | Capital greek letter gamma |
| $\delta$ | Greek letter delta |
| $\Delta$ | Capital greek letter delta |
| $\theta$ | Greek letter theta |
| $\kappa$ | Greek letter kappa |
| $\chi$ | Greek letter chi |
| $\in$ | Set membership |
| $\forall$ | Universal quantification |
| $\vec{x}$ | Vector representation |
| $\|\vec{x}\|$ | Vector length |
| $\tilde{x}$ | Median |

## CONTENTS

# 1 INTRODUCTION

## 1.1 PROBLEM AND MOTIVATION

Knowledge bases are graphs in which nodes represent real-world objects and edges represent relations between them. Due to its semantic richness, knowledge graphs have been used for different tasks such as improving the quality of the results of web search and question-answering systems.

Since the quality of knowledge bases might have a great impact on these tasks, practitioners and researchers have been working on building and maintaining them. There are different ways to build knowledge bases: manually (e.g. Cyc[1]); using crowdsourcing (e.g., Freebase[2]); or automatically (e.g., DBpedia[3] and YAGO[4]). Any strategy employed to build comprehensible knowledge bases is hardly complete and free of error (PAULHEIM, 2016). Knowledge bases might have, for instance, wrong information about entities, missing entities, attributes or relations between entities.

In this work, we are particularly interested in the problem of knowledge base completeness. More specifically, we aim to extract missing attributes (and their values) of entities from text to improve knowledge base coverage. To give a concrete example, Figure 1 illustrates the missing-attribute issue on Wikipedia Infoboxes. In the article of the Gachsaran oil field the attribute "current production oil" is a common attribute in oil fields, as can be seen on other infobox examples (Ghawar Field and Ahvaz oil field). This attribute is not present in the article's infobox but appears in the article text (See text excerpt placed above Gachsaran infobox example on Figure 1).

Wikipedia is currently a valuable source of information, and its corpus encompasses a vast number of different domains. For this reason, a lot of effort has been made to convert its content to semantic form. As highlighted in Wu and Weld (2007), although working with Wikipedia introduces some challenges on information recovering process, it also presents several features that make it ideal for extraction of information, among them are: use of unique identifiers for entities (URI); tabular summaries as Infoboxes; rudimentary taxonomic hierarchy as page lists and categories; redirection and disambiguation pages; and a vast number of articles. According to our data collection (of indexed DBpedia datasets and Wikipedia dump - both dating from October of 2016 and released in 2017) around 5.1M articles and 2.3M infoboxes are distributed over 1.1M Categories.

In this work, we are particularly interested in improving the coverage of Wikipedia infoboxes, which are tables on the top right-hand corner of Wikipedia articles that present

[1] http://www.opencyc.org/
[2] https://developers.google.com/freebase/
[3] http://wiki.dbpedia.org/
[4] http://www.yago-knowledge.org/

characteristics of the entity which the article refers to. Because of its structural nature, Wikipedia infoboxes have been used to create knowledge bases such as DBPedia and YAGO.

According to Wu and Weld (2007) infoboxes are manually created when human authors create or edit an article, a time-consuming process, and as a result, many articles have no infoboxes, and the majority of infoboxes which do exist are incomplete. To improve the coverage of Wikipedia infoboxes, we propose an approach that extracts the values from the text of entity's article on Wikipedia, using information extraction techniques (AGGARWAL; ZHAI, 2013).

Traditional information extraction approaches use machine learning along with hand-crafted features to perform extraction (WU; WELD, 2007; LANGE; BÖHM; NAUMANN, 2010). Deep learning is an alternative to this as shown by Zhang, Zhao and LeCun (2015), which have used Convolutional Neural Networks (CNN) to learn features from text and perform text classification. Instead of defining hand-craft features, deep learning strategies can learn suitable features according to the task at hand. Deep learning has shown to be very effective for many tasks, including text processing (COLLOBERT et al., 2011). For all those reasons, we apply sequence-based deep learning techniques namely Convolutional

**Source: Wikipedia. Viewed in November 27th 2018**

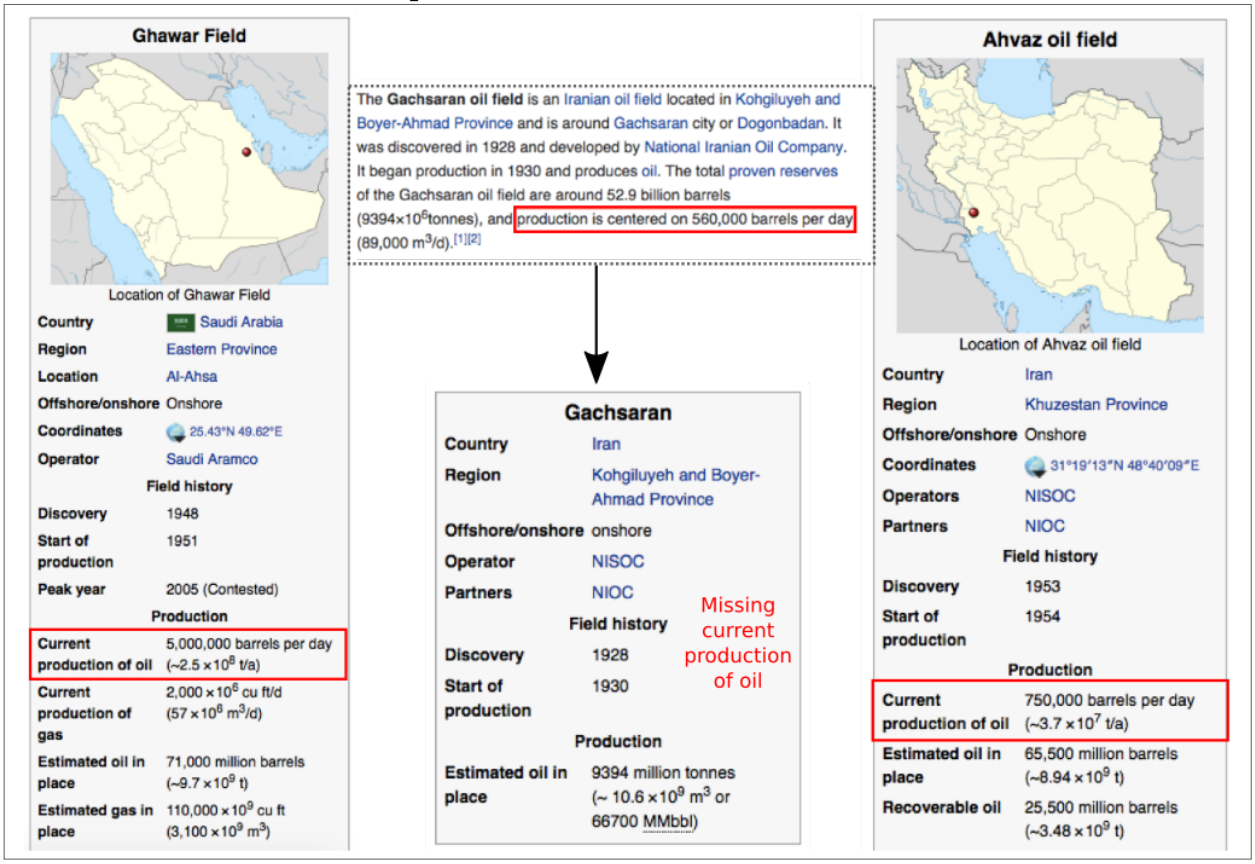



Figure 1 – Infoboxes examples showing schema drift between entities under the same domain.

and Recurrent Neural Networks to extract values of missing attributes.

To avoid having to manually label examples to train the extraction models, we apply the weakly supervised approach of Distant Supervision (MINTZ et al., 2009). More specifically we leverage the fact that, on the same Wikipedia page, characteristics of an entity are usually mentioned both on its text and its infobox. The information in the infobox is then matched with the text using a text similarity measure (Soft TF-idf) to label the tokens in the sentences in order to train the extractors.

## 1.2 RESEARCH QUESTIONS

The main goal of this research work is to provide an extraction pipeline to retrieve textual information from plain text articles and fulfill predefined schemata. This approach can be categorized as an external method for knowledge base completion. As a requirement to achieve this goal, some objectives were defined as the target for this research.

- Construct training datasets through the alignment of distant supervision with similarity measure: Is it possible to apply a similarity measure, e.g. Soft TF-IDF (COHEN; RAVIKUMAR; FIENBERG, 2003), to distant supervision as a means of dataset creation?
- Build the extraction pipeline using deep learning: Can a deep neural network architecture be competitive with traditional information extraction models in the attribute extraction task?

## 1.3 WORK ORGANIZATION

This section introduces each chapter of this work. The work is organized as follows. Chapter 2 gives an overview about main concepts, terminologies and works in the fields of Knowledge Base Construction, Information Extraction, and Deep Learning for Natural Language Processing tasks. The Knowledge Bases section comprehend the construction and enhancement of these structures, and the Knowledge Based used throught this work, the DBpedia, discussing its datasets and extraction methods. The Distant Supervision section defines the approach of the same name, and embraces some work efforts to improve it. The Information Extraction section discuss classical and current methods applied for information extraction in structured and unstructured documents, highlighting its vantages and advantages.

Chapter 3 discusses work directly related to this research. Works that have explored Wikipedia structured information for schema discovery and data analysis. It also discusses previous works that have tried to increase the coverage, or make corrections in DBpedia and other Knowledge Bases. Under this chapter are also discussed works that have mod-

eled Sequence-based Neural Networks architectures for NLP. The two baseline works used for this research are also detailed explained.

Chapter 4 describes and discusses in details the processing of the data sources explored by this work. The advantages and difficulties, and how the found problems were surpassed. It also explains in the datails the indexing strategy employed to store the large amount of data, and the framework developed to manipulate it, explaining the applied querying mechanisms for that.

Chapter 5 gives a general analysis of the data used by this work, Wikipedia and DBpedia. It explores the Infoboxes instances distribution, Infobox Templates composition, and the Properties comprised by them. It suggests the use of Wikipedia Categories for automatic schema discovery, analyzing infoboxes and subcategories as a form to infer information domains and lead autonomous schema discovery. It also conceptualize two quality indexes to analyze the composition of infobox schemata from Wikipedia.

Chapter 6 presents DEEPEX, the extractor pipeline proposed by this work. It introduces the extraction architecture and gives the details of each module, the strategies applied to build them and how they are trained. Chapter 7 presents the experiments performed to evaluate our model, comparing it with already existing systems for the same task. It gives the experimental setup, presents the metrics for evaluation, and the tests performed to validate the differences. Lastly, it concludes the study in Chapter 8 discussing the contributions, limitations and proposing future approaches to improve the work.

# 2 CONCEPTS AND TERMINOLOGY

This chapter introduces the concepts and terminologies used throughout this work. Section 2.1 introduces Knowledge Base, its importance, how they are represented, built, and enhanced, as well as its strengths and weaknesses, and also presents details about DBPedia, the knowledge base used in this work. Section 2.2 presents Distant Supervision, a strategy of automatically labelling data. Section 2.3 presents an overview of Information Extraction, showing its central concepts, techniques and models. Sequence-based models used for labeling sequenced information in the area of information extraction are presented in Section 2.4. The word representation problem is also discussed in Section 2.5, showing alternatives and techniques applied to linear and non-linear models dealing with textual information.

## 2.1 KNOWLEDGE BASES

**Source: Data extracted from DBpedia Knowledge base. Image created for this research.**

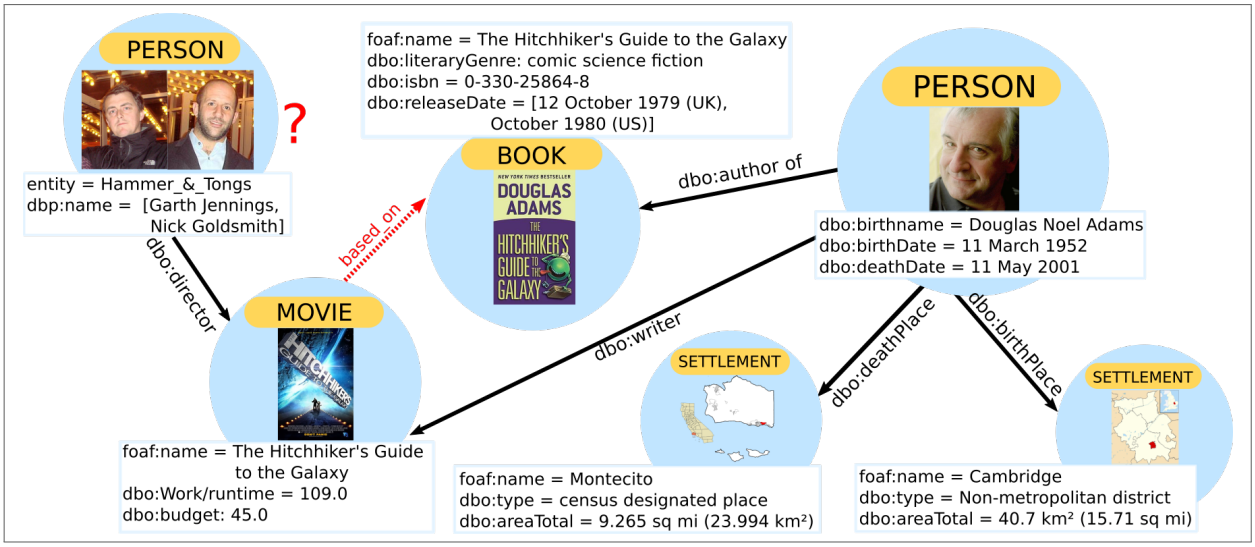



Figure 2 – Knowledge Graph snippet of DBpedia.

Knowledge bases are graphs in which nodes represent real-world objects and edges represent relations between them. These objects represented by nodes on graphs are also called Entities. According to Balog (2018), an entity is a uniquely identifiable object or thing, characterized by its name(s), type(s), attributes and relationships to other entities. Although the terms Knowledge Base and Knowledge Graph represent the same concept, Balog (2018) highlights that *"when the emphasis is on the relationships between entities, a knowledge base is often referred to as a knowledge graph."*. Figure 2 shows a snippet of DBpedia Knowledge Base (KB), how the nodes (entities) are related, and a subset of their

attributes. Knowledge graphs are important to organize the data, allowing an intuitive exploration across its structures. Besides, they are crucial for semantically explore the data meanings and consequently enable the development of tasks oriented to knowledge, i.e., it enables a better understanding and use of the available data by machines.

The information used to build knowledge graphs can come from several sources. It can be obtained from structured sources (e.g. Wikipedia Infoboxes, tables of product specification from online stores, databases, social networks, among others), unstructured texts (e.g. news articles, Wikipedia articles, any site on the World Wide Web (WWW), posts on social media, and so on), and it can also come in multimedia form (as images and videos).

Currently, the main form to represent and exchange obtained data to knowledge graphs structure are through the use of Semantic Web standards as Resource Description Framework (RDF)[1]. RDF is a World Wide Web Consortium (W3C) specification for data modeling and interchange on the Web, is currently used for content description, modeling, and knowledge management. RDF documents are composed of three main components: resource, property, and value. With these components at hand, the RDF standard can replicate existing entities links on the Web. Using Uniform Resource Identifier (URI) to represent resources, as well as the relationships between them.

### 2.1.1 Knowledge Base Construction

Due to its semantic richness, knowledge graphs have provided structured information on the Web and have been used for different tasks such as improving the quality of the results on web search (HALEVY; NORVIG; PEREIRA, 2009) and question-answering systems (FERRUCCI et al., 2010). Sa Alex Ratner and Zhang (2017) points out that the process of populating a structured relational database from unstructured sources it is of great interest for the database community. At the same time, the natural language processing and machine learning communities are pursuing the same problem under the name of Knowledge Base Construction (KBC).

To tackle the problem, KBC has been an area of intense study over the last decade, presenting different approaches. Paulheim (2017) points out that a Knowledge Graph can be built with the use of different methods. They can be curated by an organization or a small, closed group of people, through crowd-sourced effort, or created with heuristic, automatic or semi-automatic means. Sa Alex Ratner and Zhang (2017) separates these approaches in two main categories: Rule-based systems and Statistical Approaches. The first one relies on a set of rules and patterns to derive relationships, while the second tries to overcome rule-based limitations through the use of probabilistic machine learning models. Details of the categorization are described below.

[1] <https://www.w3.org/RDF/>

- **Rule-based Approaches**: In these approaches rules can lead the extraction process. However, they require that the developer builds patterns for the domain, usually through the use of string matching, heuristics and expression rules. An example of this type of approach is the "Hearst Pattern" proposed by Hearst (1992), in which was discovered that simple patterns like "X such as Y." could assist the extraction of a large number of hyponyms, e.g., it states that from the sentence *"The bow lute, such as the Bambara ndang, is plucked and has an individual curved neck for each string."* the hyponym (*Bambara ndang*, *bow lute*) can be extracted given the intuitive inferring that *"Bambara ndang"* is kind of a *"bow lute"* even if the reader is not familiar with the terms. DBLife (DEROSE et al., 2007; SHEN et al., 2007) is another example of rule-based approach. It identifies mentions of people in the database research community by performing string matching between the corpus and heuristic variations of a dictionary of canonical person names (e.g., abbreviations and first/last name ordering). Because the lack of ambiguities when dealing with this domain of information, DBLife can achieve high recall and high precision. Although, these approaches can achieve reasonable extraction performance for specific domains, their limitations make it unable to scale to large corpus or different domains. Trying to overcome the limitations of grammar-based and regular expressions extraction, Krishnamurthy et al. (2008) and Li et al. (2011) have proposed IBM's SystemT, which aids developers in the construction of high-quality extraction patterns through the use of a declarative language called AQL, which consists of a combination of the familiar syntax of SQL and some defined text-specific algebra operators. Niu et al. (2012) point out that the development process of rule-based KBC systems is increasingly being assisted by statistical techniques.

- **Statistical Approaches**: these approaches automatically select the patterns from a range of a priori features from input data, usually lexical, syntactic and semantic features. The extraction is performed through supervised learning, and candidate entities are associated with the likelihood of the instance being positive. For that are used machine learning models such as logistic regression or conditional random fields. The training examples are usually composed of annotations of sentences. As examples of systems applying this approach we have the following: Elementary (NIU et al., 2012), take as input a variety of data sources, perform information extraction with standard NLP tools and machine learning algorithms, then builds a Markov Logic Network system to perform statistical inference and make predictions for the target KB; Deep Dive (SA ALEX RATNER; ZHANG, 2017) uses a high-level declarative language based on SQL and on Markov Logic, its extraction process passes through two fases, the first evaluates the sequence of SQL queries informed to produce a factor graph of extracted information, the second step is the inference fase where statistical inferences are performed over the generated factor graph; Google's

knowledge vault (DONG et al., 2014) also combines different data sources for KBC, however, it verifies the extractions coming from Web content using existing knowledge bases, working as a knowledge fusion approach, existing KBs are used to apply a Local Closed World Assumption and fit probabilistic binary classifiers to compute the probability of a extracted tuple being true, adding to the KB only th extracted tuples with a high probability of being a true.

Beyond rule-based and statistical approaches, Niu et al. (2012) also points out distant supervision and human feedback as common techniques for KBC. In addition to these approaches we can mention the manual efforts to build KBs, which rely on human knowledge engineers to add information to the base (e.g., Cyc[2]).

### 2.1.2 Knowledge Base Refinement

Although there is a vast number of techniques to be applied in KBC, when talking about Corpus scaling to the size of the Web, it is almost impossible to achieve one hundred percent of information coverage in the initial construction. As stated by Balog (2018) *"Knowledge bases are bound to be incomplete; there is always additional information to be added or updated"*. Hence, some strategies can be used to fix inconsistencies or include missing information. This problem is managed through Knowledge Base/Graph Refinement. Paulheim (2016) groups the approaches for KB refinement into three orthogonal categories: completion versus correction, refinement target and internal versus external methods. They are detailed below:

- **Completion versus Error detection/Correction**: This categorization takes into consideration the main goal of the refinement. The addition of missing knowledge to the graph is called *Completion* and is related to the quality dimension *Completeness*. The identification of wrong information in the graph is called *Error Detection* and is related to the quality dimension *free-of-error*. As examples of appoaches on KB *Completion*, Nickel and Ring (2012) has proposed the use of matrix factorization to predict entity types in YAGO (SUCHANEK; KASNECI; WEIKUM, 2007), while Paulheim (2012) have proposed the use of association rule mining to predict missing types in DBpedia, the method is basically based on redundancies, trying to find patterns of co-occuring types to predict new ones. Aprosio, Giuliano and Lavelli (2013a) makes use of cross-language links and overlappings between category tags as features to train a supervised kernel-based classifier and predict missing types for the DBpedia ontology. As examples of *Error Detection* approaches, Paulheim (2014) represents relationships between entities in DBpedia as a high dimensional feature vector and employs outlier detection methods to identify wrong interlinks.

[2] <http://www.opencyc.org/>

Wienand and Paulheim (2014) it also makes use of outlier detection methods to detect errors in DBpedia, more specifically wrong numerical values in the KB.

- **Refinement target**: This categorization considers the targeted information. Which are the entity type, the relationship between entities, interlinks between knowledge graphs and datatypes. Nickel and Ring (2012), Paulheim (2012), and Aprosio, Giuliano and Lavelli (2013a) uses the entity types as target for enhancement, while Paulheim (2014), Wu and Weld (2007), and Lange, Böhm and Naumann (2010) make use of the relations between entities. The work of Wienand and Paulheim (2014) focuses on numerical values in the KB.

- **Internal versus External methods**: This categorization takes into consideration the provenance of the data used by the approach. Internal methods only use the knowledge graph itself. External methods use additional data, e.g., external text corpora. Nickel and Ring (2012), Paulheim (2012), Paulheim (2014), and Wienand and Paulheim (2014) are internal methods, once it makes use of only the graph information itself, while Aprosio, Giuliano and Lavelli (2013a), Wu and Weld (2007), and Lange, Böhm and Naumann (2010) are external methods which make use of external sources like cross-language editions of the DBpedia an unstructured sources.

As aforementioned, there are different ways a knowledge base can be built and enhanced: manually, using crowdsourcing or automatically. While manually and crowdsourcing KB rely on human involvement and curation, automatic approaches mainly make use of knowledge extraction and graph construction with the assistance of handcrafted patterns and machine learning models. The knowledge extraction, used both for construction and enhancement of KBs, can be performed on structured, semi-structured, and unstructured documents, mostly on the web, given its extensive amount of data and different sources.

The graph construction is related to the completeness of already existent KBs. This perspective makes use of the graph information itself or other graphs, without external corpus or extractions. These approaches try to predict new entities or relationships between them. Named entity found by extraction methods can be highly ambiguous, to build consistent Knowledge Graph (KG)s it is necessary to apply methods like entity resolution, entity linking, and link prediction tasks. Entity resolution consists in the distinction of entities with the same name and different names for the same entity, to fix inconsistent information on KG. Entity Linking consists in to determine the identity of extracted entities, it differs from Named Entity Recognition (NER) because NER only identifies occurrences of an entity in the text, Entity Linking tries to link those identified occurrences in the text to its corresponding entries in a KG. The Linking prediction is used to predict possible future links in the graph. It can also be used to predict missing links between related entities due to incomplete data.

Any strategy employed to build and maintain comprehensible knowledge bases is hardly complete and free of error. As previously shown, Knowledge bases may have, for instance, wrong or inconsistent information about entities, missing entities, attributes or relations between entities. In this work, we are going to explore the completeness of these structures through Information Extraction in unstructured sources. DBpedia Knowledge Base, the KB used in this work, makes use of ontology mappings to annotate structured Wikipedia sources, and then extract information. The next subsection will focus on this Knowledge Base, its composition, and latest release.

### 2.1.3 The DBpedia datasets

The DBpedia community project presented on Lehmann et al. (2015) builds a large-scale, multilingual knowledge base by extracting structured information from Wikipedia. The latest DBpedia version (2016-10, publication year: 2017) describes 6.6M entities consisting of 1.5M persons, 840K places, 496K works, 298K organizations, 306K species, 58K plants, and 6K diseases. The total number of resources described only in English Wikipedia is 18M.

The extracted information is set in the form of RDF statements and served as Linked Open Data (LOD) on the Web. The project makes use of a community effort to build mappings from Wikipedia structured information representation to a DBpedia ontology. This approach makes possible a consistent extraction of high-quality data and datatype information (e.g.: geographic, date-temporal, and numbers). DBpedia provides a variety of datasets organized in RDF format. They are mainly built using extractors divided into four categories:

- **Mapping-based Infobox Extraction**: It maps infobox properties and values to manually written mappings between them and terms in the DBpedia ontology, which is a shallow, cross-domain ontology, and has been manually created based on the most commonly used infoboxes within Wikipedia, currently it covers 685 classes described by 2,795 different properties[3]. This approach extracts high-quality data and provides datatype definition for each infobox property. However, the extracted information is restricted to ontology mapped structures;

- **Raw Infobox Extraction**: Existing infobox information is extracted and directly mapped to RDF statements. This extraction does not rely on manual mappings to ontologies and does not provide datatype definition for properties. Hence, the quality of the extracted data is poor.

- **Feature Extraction**: Specialized extractors to get single article features, e.g. geographic coordinates;

[3] <https://wiki.dbpedia.org/services-resources/ontology>

- **Statistical Extraction**: It extracts information from the unstructured text to provide data based on statistical measures of page links or word counts. For instance, the grammatical gender dataset tries to identify the gender of a given resource mapped to the class "*dbo:Person*" on DBpedia ontology.

## 2.2 DISTANT SUPERVISION

Distant Supervision (DS) is a paradigm usually applied by Relation Extraction methods to build training datasets. Instead of relying on human handcrafted features, patterns or manual annotation of training examples, DS starts from the assumption that any sentence containing a pair of entities from a known relation is likely to express that relation. Mintz et al. (2009) have used Freebase aligned with 1.2 million Wikipedia articles to generate training sets of 102 Freebase relations and entity pairs that participate in those relations.

Takamatsu, Sato and Nakagawa (2012) highlights, that although DS is an attractive approach to heuristically generate a large number of labeled data, when compared with the limitations of supervised approaches, it can generate noisy labeled data and cause poor extraction performance. It states that this can happen when the given entity pair express more than one relation on target text. As an example, in the context of Relation Extraction, the pair (Michael Jackson, Gary) expressing *place_of_birth* relation on the KB might be matched with the sentence "*Michael Jackson moved from Gary.*" which does not rightfully represents the relation *place_of_birth*. On the other hand, the sentence "*Michael Jackson was born in Gary*" is a good representation for the relation. Roth et al. (2013) have organized DS approaches into three basic principles:

- At-least-one constraint: it considers that at least one sentence labeled positive by the DS assumption actually represents a true positive sample (RIEDEL; YAO; MCCALLUM, 2010; HOFFMANN et al., 2011; SURDEANU et al., 2010);
- Topic-based models: makes use of a generative model to discriminate between patterns that are expressing the relation and ambiguous ones (ALFONSECA et al., 2011);
- Pattern correlations: make use of a probabilistic graphic model containing hidden variables to model whether a pattern expresses a relation or not (TAKAMATSU; SATO; NAKAGAWA, 2012).

## 2.3 INFORMATION EXTRACTION

Most Information Extraction (IE) methods rely on NER, a sequence labeling problem where categories should be applied to each term. Balog (2018) points out that the main technique to train an NER model is to use a large labeled corpus. It also highlights that sequence labeling models widely used are hidden Markov models (ZHOU; SU, 2001) and Conditional Random Fields (FINKEL; GRENAGER; MANNING, 2005).

As pointed out by Weld and Hoffmann (2008), IE techniques has received a lot of attention in latest years. IE has showed potential in convert natural language texts on the web to relational form. Consequently, it has made possible the discovery of textual knowledge, data integration, the use of advanced searches, development of question-answering systems, text summarization and even ontology learning.

Grishman (1997) defines IE as *"the identification of instances of a particular class of events or relationships in a natural language text, and the extraction of the relevant arguments of the event or relationship. [...] involves the creation of a structured representation (such as a database) of selected information drawn from the text."*. IE can extract information from unstructured, semi-structured or structured data. However, recently the effort has been pointed to extract structure from unstructured information, e.g., natural language texts, on a large scale.

This section presents an overview about approaches used for Information Extraction. It embraces classical methods until recent ones, emphasizing its positives and negatives points.

### 2.3.1 Approaches on Information Extraction

Beyond syntactic parsing and sequence tagging for entity extraction, traditional IE methods have relied on supervised or bootstrapping approaches for relation extraction. Bootstrapping techniques have used relation specific pattern examples, as sources to induce rules and extract information (RILOFF, 1993; SODERLAND et al., 1995). The two main approaches for information extraction are rule and feature-based. Rule-based methods apply linguistic rules to capture patterns typically used to express relations, as in Agichtein and Gravano (2000), and Brin (1999). Feature-based approaches use machine learning technics to learn lexical, syntactic and semantic features from input data and extract target relations through supervised learning (GUODONG et al., 2005).

These traditional methods rely on sample patterns or on supervised learning, to learn syntactic and semantic features on training data on a given corpus, as a consequence, it is required the specification of a target schema to lead the extraction process. As shown by Vo and Bagheri (2016), these characteristics imply on limited scalability and portability across domains, which, e.g., can be challenging to apply to a large corpus as the Web.

A new paradigm for information extraction is proposed by Banko et al. (2007), the Open Information Extraction (OpenIE), it intends to work with an unlimited relation number and run fast enough to process the diversity of corpus on the Web. This paradigm excludes human involvement in rule definition or training examples labeling. Thus, OpenIE systems apply self-supervised learning where automatic heuristics are responsible for generating labeled data to feed extractors, and target relations are extracted automatically. As pointed out by Banko et al. (2007), OpenIE is an extraction paradigm that facilitates domain-independent discovery of relations extracted from text and readily

scales to the diversity and size of the Web corpus. However, Vo and Bagheri (2016) has shown that extractions using this approach usually present incoherent or uninformative relations.

Alfonseca et al. (2011) also highlights the existence of unsupervised methods in Relation Extraction (RE), they are called unsupervised semantic parsing. The main objective of these approaches is to use inference over semantic representations of text, obtained through the clustering of mentioned entities and relations. Although they are powerful models, they also require high computational resources.

The DS method, proposed by Mintz et al. (2009) and presented on previous sections, comes up as an alternative for the approaches aforementioned. This method can be categorized as a weakly supervised approach, where the model can heuristically label training examples using existing structured information. Looking for scalability and portability across domains, the work presented here intends to apply distant supervision as a weakly-supervision method to build training datasets for text classification and entity extraction. In spite the fact that the DS assumption is mostly applied in a Relation Extraction perspective, our goal is to use the assumption for NER tagging of sentence tokens. Identifying entities occurrences in the natural language text through the matching of value and properties existing in the structured source.

## 2.4 SEQUENCE-BASED MODELS

Sequence classification can be applied to a variety of fields as genomic analysis, information retrieval, finances, time series prediction and information extraction. Xing, Pei and Keogh (2010) states that a sequence is an ordered list of events and may carry a sequence of labels, in the case of natural language processing consists in labeling each element in the sequence, e.g., assigning a category for each word in a sentence, such as name identity, noun phrase, verb phrase etc. Xing, Pei and Keogh (2010) also highlights that the sequence labeling problem has been solved by using conditional random fields. However, other models have beem applied such as combining Hidden Markov Model (HMM) and SVM, and using recurrent neural networks

This section presents the main sequence-based models applied for Information Extraction tasks. In the first subsection is shown the Conditional Random Fields, an advanced implementation of Markov Random Fields, widely used for the task. The second subsection presents Sequence-based Neural Network models, a novel approach for deal with Information Extraction ad Natural Language Processing.

### 2.4.1 Conditional Random Fields

Conditional Random Fields (CRF) is a sequence modeling framework commonly used for Natural Language Processing (NLP) tasks. It was formalized by Lafferty, Mccallum

and Pereira (2001), and is defined as *"a framework for building probabilistic models to segment and label sequence data"*. The CRF models P=($\vec{y}|\vec{x}$) using a Markov Random Field, assuming that $\vec{x}$ and $\vec{y}$ have the same length. The nodes of the Markov Random Field corresponds to elements of $\vec{y}$ and features that are conditional on $\vec{x}$. The training is done to maximize the likelihood of ($\vec{x}$, $\vec{y}$) pairs from training data.

According with Sutton (2012), CRFs have been applied to many problems in natural language processing, including named-entity recognition (MCCALLUM; LI, 2003), shallow parsing (SHA; PEREIRA, 2003), word alignment in machine translation (BLUNSOM; COHN, 2006), citation extraction from research papers (PENG; MCCALLUM, 2006), extraction of information from tables in text documents (PINTO et al., 2003), and many others. Directly related with this research are the works of Wu and Weld (2007), and Lange, Böhm and Naumann (2010) which have applied CRF aligned with engineered features to extract information through NER tagging. The main drawback in CRFs to model sequential inputs to outputs is the need for feature engineering definition. Works applying this model have relied on the specification and modeling of syntactic features.

### 2.4.2 Sequence-based Neural Networks

The Sequence-based Neural Network is a newly approach to perform the task of sequence classification using Deep Neural Network. Sequence-based Neural Network models have been applied to sentiment classification, semantic relation classification, question answering, syntactic parsing, sequence tagging, and character level text classification.

The basic unit of these mechanisms is the neuron. It requires scalar inputs with associated weights. A multiplication operation is applied and is followed by a sum of its elements. A nonlinear function is applied to the result which is passed to an output layer. A Feed-forward Neural Network is the combination of these Neurons, which are organized in layers of a network where the output of a neuron can be the input of one or more neurons. When all neurons of a layer are connected through weights with all neurons in the next layer, this is called a fully connected layer. Deep Learning refers to more advanced architectures of neural networks. Deep learning presents several hidden layers and not all neurons must be connected. The main deep learning models are CNN and Recurrent Neural Networks (RNN).

The building of a CNN architecture for text classification is pointed out by Zhang and Wallace (2015) as starting by converting a sentence into a sentence matrix where the rows are word vector representations of each token. These word representations can be trained from available models as word2vec (MIKOLOV et al., 2013) and GloVe (PENNINGTON; SOCHER; MANNING, 2014). Different from feed-forward neural networks, CNNs do not apply weights connecting neurons on different layers. CNNs make use of convolutions to compute the output. Convolutions are functions applied over the input matrix. A filter is used to segment the matrix in different regions and learn features from it. Each hidden

layer on the CNN apply different filters to the previously resulted matrix, at the end, the convolution results coming to the output layer are combined to get the classification.

CNNs have mostly been successfully applied to computer vision tasks, but it has also overcome state-of-art results for tasks as speech recognition, handwritten characters classification, and more recently it has been applied for natural text classification. Santos, Xiang and Zhou (2015) have proposed the use of CNN to classify relations through ranking of loss function. They could outperform state-of-the-art results using the SemEval-2010 Task 8 dataset without using handcrafted features. Santos, Xiang and Zhou (2015) have used CNN to learn features from sentences and do relation extraction. They have applied different window sizes, pre-trained word embeddings, and a non-static architecture to achieve the goal. Obtained results have also outperformed classical approaches for relation extraction. Zhang, Zhao and LeCun (2015) have applied CNNs to learn character level features from words and use it for text classification. This approach does not require knowledge about the syntactic or semantic structure of a language making it easy for a system using it to work with different languages.

The RNN model was first proposed by Elman (1990) and consists of a neural network that map input sequences to output sequences. This network presents an internal state that stores context information, allowing it to keep information about past inputs for an amount of time that is not fixed a priori. RNNs have been successfully applied to solve problems like speech recognition, language modeling, language translation, image captioning, and time series prediction. Although RNNs are expected to store context information, there are cases when more context is needed. Usually when the gap between stored information and the present moment is small RNNs work successfully. Long Short-Term Memory (LSTM) is a special case of RNN and was first introduced by Hochreiter and Schmidhuber (1997). LSTM makes use of memory cells and gating mechanisms allowing the storage of information for long periods, circumventing the RNN long-term dependency for some cases.

As of the act of looking past context to predict current state, Schuster and Paliwal (1997) present a Bidirectional Recurrent Neural Networks (biRNN) which looks both ways, past, and future contexts, to better predict current state. Using a combination of both biRNN and LSTM, Graves and Schmidhuber (2005) have presented a Bidirectional Long Short-Term Memory (BLSTM) approach which has been shown good results for NER tasks (CHIU; NICHOLS, 2016).

## 2.5 WORD REPRESENTATION

Dealing with textual data is a challenging task. As stated by Goldberg (2017) most features in natural language represent discrete, categorical features such as words, letters, and part-of-speech tags. The first focus when dealing with Machine Learning tasks is how to represent these symbols, once they require well-specified input and output of fixed-

length. Hence, Machine Learning (ML) models cannot handle raw text directly, being necessary its conversion to vectors of numbers.

Strategies to represent textual information are mainly divided into two categories: One-hot representation (or sparse vectors) and feature embeddings (or dense representation). Each technique has pros and cons, recommended to be used for specific context, tasks, and different problem complexities. Goldberg (2017) accentuate these differences.

In the One-hot encode approach, each dimension corresponds to a unique feature, i.e. each word or letter in the vocabulary is treated as a single vector with the same size as the vocabulary, the word or letter itself is identified by a unique value 1 (one) indicating its position in the index, all other positions are filled with value 0 (zero). This representation is also called sparse vectors, since it requires more dimensions to represent vocabulary information and all entries are zero except the single entry corresponding to the respective word/letter.

Another widely used sparse representation for the feature extraction procedure is the Bag-of-Words (BoW). It considers each word count as a feature. Similar to one-hot encoding, the vector dimension for each word is defined by the size of the vocabulary, but it differs in the method to populate the vector. The BoW approach looks for the histogram of the words/letters within the document to fill the vector representation. Usually, this method is aligned with the Term Frequency-Inverse Document Frequency (TF-IDF) measure, to balance the scoring between occurrences of rare and highly frequent words.

In the Dense representation each core is embedded into a $d$ dimensional space. This dimension is usually lower than the number of features. Goldberg (2017) points out that the main difference is that on sparse representations each feature is its own dimension while in the dense representation each feature is mapped to a vector. The use of dense and low-dimensional vectors reduces computational effort from deeper non-linear models. However, some aspects can be considered when choosing the word representation: Dense vectors allow the model to share statistical strength between events, because of its capability to capture similarities, increasing its generalization power; one-hot representation can be used when there are distinct features and is believed that no correlations exist between different features, also when its not desired the share of information between distinct words.

# 3 RELATED WORK

The effort to build a large-scale knowledge base mainly using semi-structured information harvested from the Web usually presents inconsistencies as incompleteness and incorrect entities attributes, types or relationships with other entities. This section presents a brief overview from previous approaches that have tried to circumvent this problem. The main topics explored by these studies is Knowledge Base completeness, more specifically approaches on Information Extraction in the Wikipedia, and DBpedia data analysis for schema discovery.

## 3.1 WIKIPEDIA STRUCTURES EXPLORATION

Previous works have used Wikipedia structure to extract semantic relations, to provide answers to queries, and to identify common schemas. Nguyen et al. (2010) have explored the Wikipedia structure as infoboxes and Links between pages to conceptualize WIKIQUERY. WIKIQUERY has as primary goal to capture relations between Wikipedia entities and provide a multi-document answer to queries. To achieve this goal it states that each value of an infobox tuple is either an atomic value or a reference to another infobox, usually represented by hyperlinks. Then, it makes use of a graph structure to navigate in these relationships and returns answers to queries as a set of minimal trees from this graph. Hence, it links entities through references over its content to provide multi-document answers to queries.

The proposed strategy of Nguyen et al. (2012) is to cluster infobox structured information and discover entity types. To address this problem it is defined a clustering strategy called WIClust (Wikipedia Infobox Clustering), an unsupervised method that receives as input a set of infoboxes, applies correlations among its properties to identify the important ones and uses it as the base for the cluster. The output is a set of entity types represented by each resulted cluster. This method does not require the number of clusters (or types) to be known in advance. However, they do not make use of any other type of structured information on Wikipedia beyond infobox instances. Different from these works, the study shown here makes use of other Wikipedia structures beyond infobox instances, e.g., Categories hierarchy, and infobox templates. The main goal is to support the schema discovery process through data analysis and proposed quality measures. This initiative may assist the definition of entity types in a general form, focusing on the information domain, and without require the input of a set of infobox instances to perform clustering.

Mohamed and Oussalah (2014) have made use of Infobox templates and instances to increase the number of training examples for named entity classification and recognition

problems. However, despite the use of Wikipedia structures, its focus is gather training examples to increase coverage and represent named entities for Person, Location and Organization in NER problems, without autonomous Infobox creation.

## 3.2 KNOWLEDGE GRAPHS ENHANCEMENT

Works have tried to enhance existing knowledge graphs. Trying to circumvent the problems of incorrect or incomplete information about entities, relationships, types, and literal values. Works as Paulheim and Bizer (2013), Paulheim and Bizer (2014), Sleeman and Finin (2013), Nickel and Ring (2012), and Sleeman, Finin and Joshi (2015) have applied internal methods for Knowledge Graph refinement, using only the graph information itself to infer and add missing knowledge or identify erroneous pieces of information. Nickel and Ring (2012) have focused in incorporate ontological knowledge to a factorization sparse vector for learning relations of the YAGO Knowledge Base. The other cited works have also made use of statistical and machine learning approaches, as probabilistic and topic modeling, but they have mainly focused on the DBpedia knowledge graph.

DBpedia makes use of four types of extractors, as shown in the previous chapter. Aprosio, Giuliano and Lavelli (2013c) states that the problem of coverage on DBpedia data is mainly caused by the lack of existing Infobox instances for some pages, and by the limitation of mappings for its ontology. Previous works (ZAVERI et al., 2013; FLEISCHHACKER et al., 2014; WIENAND; PAULHEIM, 2014; PAULHEIM, 2017) have analyzed and tried to measure the quality of DBpedia data and ontology mappings, mostly through the use of outliers detection and analysis of ontology mappings. Font, Zouaq and Gagnon (2017) have proposed the use of OpenIE and information already available on the knowledge graph to partly correct existing problems. Different from them, we have performed a joint data analysis on DBpedia datasets and Wikipedia structured information, relating infobox structure to Categories hierarchy in order to lead an automatic schema discovery.

Aprosio, Giuliano and Lavelli (2013a), and Aprosio, Giuliano and Lavelli (2013b) tries to overcome the coverage problem through automatic mapping of infobox templates to DBpedia ontology. It makes use of a rule-based approach to map already existing Wikipedia infoboxes, taken from different language versions of Wikipedia pages, and maps to the most probable class in the ontology. The limitation of the work is the mapping of only existing infoboxes, while there is a large number of entities with infobox missing information or without infobox instances. The work here proposed tackles the problem through natural language processing, to fill existing infoboxes or create new ones.

## 3.3 INFOBOX INSTANCES FOR KNOWLEDGE BASE ENHANCEMENT

The automatic creation or update of Infobox instances is constantly explored as a means of extend Knowledge Bases information coverage. The automatic creation or update can

come from various approaches, between them we can highlight the ones using Semantic Web, Relation Extraction, Entity Extraction, and already existing Knowledge Bases.

The approach presented by Banerjee and Tsioutsiouliklis (2018) shows a method for Relation Extraction exploring Wikipedia corpus and structures. It have faced the wrong labeling problem of Distant Supervision using DBpedia triples and text from articles in Wikipedia. To circumvent the problem, it has made use of confidence values provided by co-ocurrence statistics of dependency paths. The confidence values are provided as weights to proposed training model, which is a Multi-Encoder Model with three LSTM layers that encode features from Words, POS tags and dependency paths. The output of the hidden states is used to predict the relation.

The work of Sáez and Hogan (2018) derives statistics from Wikidata, an existing central hub for storage of the structured data from Wikipedia and other Wikimedia pages, to create Infobox instances. It ranks and prioritizes atribute-value pairs from entities, using it to create Wikipedia infoboxes. It does not use any other type of information like manually-specified templates, training data, or unstructured information.

Infoboxer, a tool grounded in semantic web technologies is proposed by Yus et al. (2014) for the task of infoboxes creation and updating. It identifies popular properties in infobox instances inside a category and make use of its frequencies to rank the most used. It also makes use of the DBpedia Ontology to identify attribute value types. The property values are manually informed by the user. However, when the value type is a semantic class, it makes use of the most used instances in DBpedia to suggest possible values to the property.

Different from Infoboxer (YUS et al., 2014), Kylin (WU; WELD, 2007) and iPopulator (LANGE; BÖHM; NAUMANN, 2010) are sytems directly related with the work proposed in this paper. Infoboxer is a tool to assist users during infobox creation and editing, while Kylin and iPopulator focus on autonomous infobox generation through identifying Entity properties and values for information extraction. Also, the work of Sáez and Hogan (2018) do not apply information extraction in natural language texts as a means of infobox creation, instead it generate derivates from Wikidata structured information. Kylin and iPopulator are detailed in the following subsections.

### 3.3.1 Kylin

Directly related with the work presented here, Kylin was proposed by Wu and Weld (2007) and is a system prototype that looks for pages with similar infoboxes, defines common attributes, create training datasets, and trains CRF extractors. It looks for autonomously creation and completeness of Infoboxes, as well as automatic link generation for identified nouns. Our system has focused on the first goal, autonomous creation or completeness of infoboxes. In spite of the modules from both works being similar, they are different in methods and techniques applied to achieve the goal. Also, given constant updates of

Wikipedia texts, pages, and structures, Kylin may be obsolete to the current context. The architecture of Kylin's infobox generator is composed of three main components (see Figure 3); they are described below. Details of the architecture of our system are showed in Chapter 6.

**Source: (WU; WELD, 2007)**

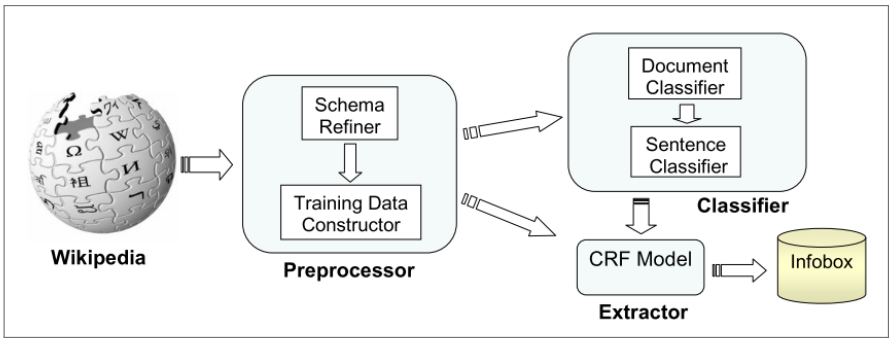



Figure 3 – Architecture of KYLIN's infobox generator

(i) Preprocessor: it selects and refines Infoboxes, choosing relevant attributes and generates training datasets for machine learning models;

(ii) Classifier: Two types of classifiers are modeled, the first is a document classifier that predicts if an article belongs to some class, the second is responsible for predicting if a sentence holds the value to a property; and

(iii) Extractor: for each property of the defined schema, one CRF model is trained to extract values for it.

Kylin makes use of strict heuristics to match between property-value and sentence tokens. Wu and Weld (2007) justifies the choose for strict heuristics, consequently building incomplete datasets, to the increase of precision over recall. Our work makes use of matched tokens similarity scores inside a sliding window of sentence tokens, following a different path. The intuition is to relax sentence classifiers to catch unusual sentences, that may contain important values, letting the extractor in charge of define what is going to be extracted. Our approach allows better extractions for unpopular categories, articles, and sentence patterns that do not happen frequently. The are also differences in classifiers and extractors models and training, that are better discussed in Chapter 7.1.2.

### 3.3.2 iPopulator

Directly related to this work, iPopulator was proposed by Lange, Böhm and Naumann (2010). It presents a naive process to extract information from plain text and populate infobox instances. Different from Kylin, iPopulator does not define the schema to be

filled, it makes use of all suggested attributes from evaluated template types, and does not apply classifiers to filter irrelevant sentences. Also, iPopulator do not make use of the entire Article text when looking for extractions, it states that the majority of the information can be found in the few first articles paragraphs.

As stated before, to build training datasets Kylin applies a set of heuristics, pursuing exact matches betweeen Infobox property-values and text sentences. Nevertheless, as asserted by Lange, Böhm and Naumann (2010) in iPopulator, often times an attribute value and its occurrence in the text do not precisely match. Based on this assumption, iPopulator define the similarities measures based on two functions (one for numerical values and the other for string values) and the Levenshtein edit distance, applying fuzzy matching to detect matching from articles text tokens and attribute values.

iPopulator process is composed of the *Structure Analysis* module, *Training Data Creation*, *Value Extractor Creation*, and *Attribute Value Extraction*. The extraction process is shown in Figure 4. Its modules are described below.

**Source: (LANGE; BÖHM; NAUMANN, 2010)**

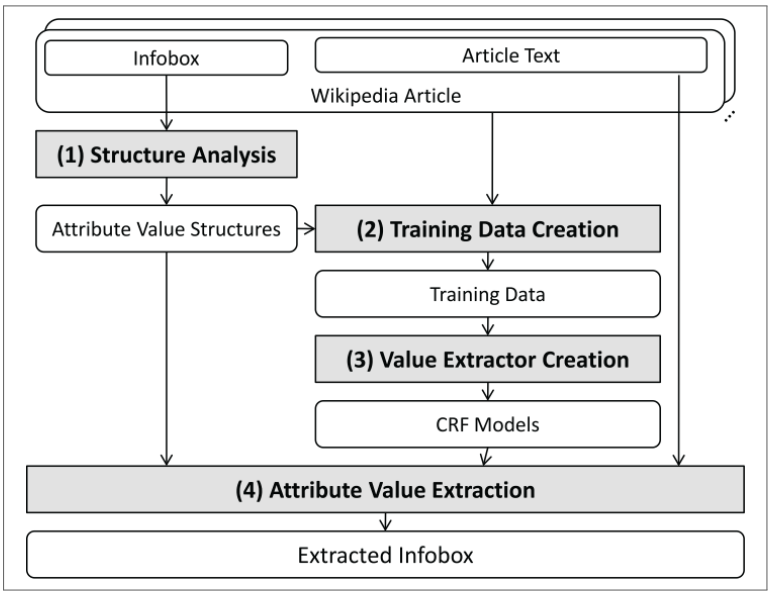



Figure 4 – iPopulator extraction process

The *Structure Analysis* module makes use of regular expressions to model a structure discovery algorithm. The algorithm is composed of parsing, counting, sorting, and merging. First, the values of each Infobox attribute are parsed, and patterns representing the syntactical structures of these values are extracted, i.e., the attribute number_of_employees from infobox_company might contain the value "12,500 (2003)", this value is parsed, and the pattern **(Number '(' Number ')')** is obtained. Afterward, obtained patterns for this attribute are counted and sorted. Important patterns, according to its frequencies, are merged to obtain a unique representative pattern for the attribute. Some rules are applied to identify optional elements of the structure, as well as lists recog-

nition (an attribute is composed by multiple values of the same type, forming a list. e.g., band members and film actors).

For *Training Data Creation*, iPopulator restricts the size of Articles text to only a few first paragraphs, using the first five paragraphs in executed experiments. Sentences are aligned to fuzzy matching for the labeling task. Parts of the value structure previously defined are used as labels for sequence tagging. The fuzzy matching is composed of two functions, differing between textual and numeric values. The *Value Extractors Creation* is made with the training of CRF models. One extractor is trained for each template attribute, using the previously built training datasets as input.

The *Attribute Value Extraction* applied by iPopulator make use of three basic heuristics to lead information extraction. First, the order of extracted tokens is aligned to the obtained value structure and the position of the tokens. Second, structural elements as commas or brackets are not directly extracted from the text. Instead, they are kept from previously obtained value structure. Third and last, meaningless values are removed from extraction, e.g., optional tokens (marked with **?** in obtained structure pattern) are only removed if preceded by mandatory tokens.

## 3.4 SEQUENCE-BASED NEURAL NETWORKS FOR NLP

RNNs are shown by Goldberg (2017) as producing strong results in language modeling, sequence tagging, parsing, and the modeling of character sequences and part-of-speech tags. As a practical application of RNNs, Chiu and Nichols (2016) have designed architecture of network merging word and character features as well as pre-trained embeddings of words. They have conceptualized the network as a hybrid of BLSTM, to model the sequence of words backward and forward, along with a CNN to capture character level features. The architecture has overcome state-of-the-art results on the OntoNotes 5.0 and CoNLL-2003 datasets. As highlighted by Chiu and Nichols (2016), the classical systems on NER rely on heavy feature engineering, proprietary lexicons, and rich entity linking information.

# 4 DATA DESCRIPTION AND PROCESSING

The data used by this work comes from a combination of DBpedia (LEHMANN et al., 2015) datasets and extractions from raw Wikipedia Dump. DBpedia is a knowledge base represented by RDF triples, created from Wikipedia structured information and following mapped ontologies. It provides information about entities, categories which the entities belong to, hyperlinks between entities, subcategories, among others. For this study, we used datasets from the English version of the dump of October of 2016[1] and released in 2017. The Wikipedia Dump is a complete copy of all Wikipedia pages, in the form of Wikitext source and metadata embedded in Extensible Markup Language (XML) format. The Dump used in this study dates from October 2016, and is the same used in the extraction of the DBpedia datasets.

From DBpedia we have used datasets that provide information about Categories, Templates and natural text from Articles. Wikipedia Dump has been used for the extraction of Infobox instances. This combination was due to DBpedia coverage problem on Infobox information. To efficiently query this information and be able to generate statistics for this study, we have used Apache Lucene[2]. Apache Lucene is a free and open-source information retrieval library and has been widely recognized for its utility in the implementation of search engines being suitable for applications requiring text indexing and searching capability. Therefore, this work applies basic indexing and searching strategy of DBpedia datasets and Wikipedia DUMP through Apache Lucene capacity.

In the remaining of this chapter, we provide details about the data and our strategy to process it for the analysis. To make easy the understanding of concepts, section 4.1 contains a glossary of Wikipedia structures used by this work. Section 4.2 presents the indexing mechanism applied to provide manipulation of Wikipedia structures as Categories, Subcategories, and Template parameters. Section 4.3 shows the same indexing strategy applied to Infoboxes. The indexing of Wikipedia article texts is described in Section 4.4. The querying schema to recover indexing information is shown in Section 4.5.

## 4.1 WIKIPEDIA STRUCTURE

Wikipedia presents a vast structure to organize its information. Below are listed and described the main structures explored by this work:

- Categories: group together pages on similar subjects;
- Subcategories: represents an *IS-A* relation between categories, implying that if Category *B* *IS-A* Category *A*, articles under *B* are also directly related to Category *A*.

[1] <http://wiki.dbpedia.org/datasets/dbpedia-version-2016-10>
[2] <https://lucene.apache.org/core/>

**Source: Wikipedia. Viewed in January 30th 2019**

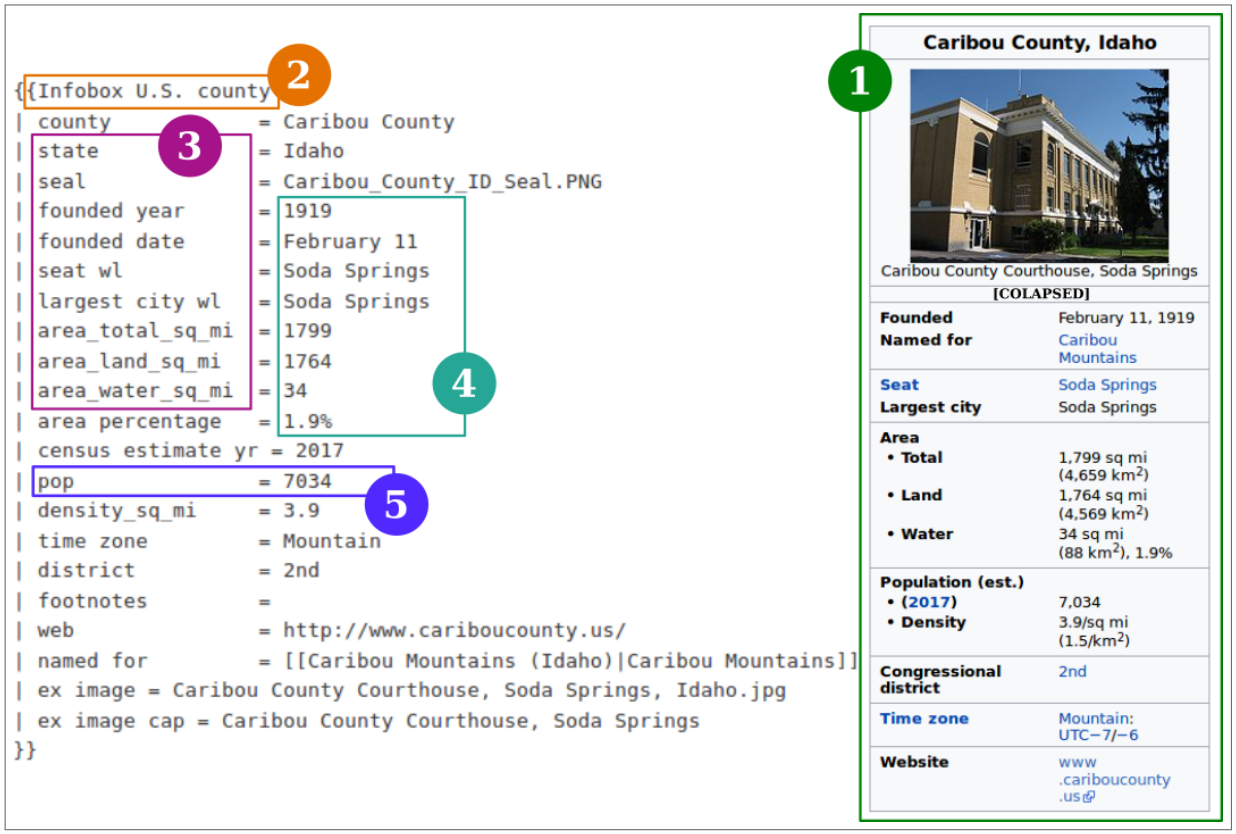



Figure 5 – Wikicode (left) example to render Infobox (right) on Wikipedia page

The subcategorization makes it possible organize categories. It is an acyclic directed graph that has the categories as vertices and the parent-child relationships as edges;

- Infobox (instance): is a fixed-format table on articles that present concise and relevant information to the topic. Number 1 in Figure 5;
- Infobox Template[3]: Also called of Infobox Type and Infobox Class, are templates that provide standardized information across related articles. It suggests properties to be used when filling information. Nonetheless, their use is not mandatory. An Infobox type must be specified during Infobox creation or editing. The user should inform it at the infobox mapping header (see Number 2 in Figure 5);
- Property or Attribute: a parameter used to identify an aspect to be filled when creating or editing Infoboxes. E.g., name, birth date, and nationality. Number 3 in Figure 5;
- Property value: value assigned to a property. Number 4 in Figure 5;
- Tuple: a set of one property and its value(s). Number 5 in Figure 5.

[3] <https://en.wikipedia.org/wiki/Help:Infobox>

Listing 4.1 – Tuple examples for each dataset along with its respective processing output.

```
(
    <http://dbpedia.org/resource/Tow_Law>
    <http://purl.org/dc/terms/subject>
    <http://dbpedia.org/resource/Category:Wind_farms_in_England> .
)
Indexing Terms:
Subject = Tow_Law
Predicate = subject
Object = Wind_farms_in_England
```

(a) article_categories_en.ttl is the original dataset name for **article-categories**

```
(
    <http://dbpedia.org/resource/Category:Wind_farms_in_England>
    <http://www.w3.org/2004/02/skos/core#broader>
    <http://dbpedia.org/resource/Category:Wind_farms_in_the_United_Kingdom>
)
Indexing Terms:
Subject = Wind_farms_in_England
Predicate = none
Object = Wind_farms_in_the_United_Kingdom
```

(b) skos_categories_en.ttl is the original dataset name for **skos-categories**

```
(
    <http://en.dbpedia.org/resource/Template:Infobox_UK_place>
    <http://en.dbpedia.org/property/templateUsesParameter>
    "latitude" .
)
Indexing Terms:
Subject = Infobox_UK_place
Predicate = templateUsesParameter
Object = latitude
```

(c) template_parameters_en.ttl is the original dataset name for **template-parameters**

**Source: This research**

## 4.2 INDEXING WIKIPEDIA CATEGORIES AND TEMPLATES

The DBpedia community provides datasets in two serialization types: turtle (.ttl) and quad-turtle (.tql). The turtle serializations are composed of a series of data in n-triple format *(<subject> <predicate> <object>.)*, while the quad-turtle adds context information to every triple, the fourth component contains the graph name and provenance information on each triple. The quad turtle format is *(<subject> <predicate> <object> <graph/context>.)*. We restrict our approach to use turtle (.ttl) serialization datasets once the current work does not need graph context, but only the information itself.

For better readability, the selected datasets are going to be referred as **article-categories**, **skos-categories**, and **template-parameters**, respectively. Listing 4.1 shows an example of triple for each file. The selected datasets are described below.

- **article-categories** holds relations between articles and categories (or subcategories). The triple structure is organized as follows: the subject corresponds to an

Article URI, the predicate is a property and the object corresponds to the category URI. This dataset contains a total of 23,990,512 triples;

- **skos-categories** represents relations between categories and subcategories, consisting of 6,083,029 triples. This dataset uses the Simple Knowledge Organization System (SKOS) property broader namespace [4] to indicate a subcategory-category relation in the triple. The subject corresponds to the subcategory, the predicate holds the broader annotation and the object corresponds to a category;

- **template-parameters** represents the structure of infobox templates. Each triple consists of the template URI (subject), URI for the ontology property *templateUses-Parameter* (predicate) and the suggested template parameter name (object). This dataset comprises a total of 776.554 triples.

We index the DBpedia datasets iterating over each dataset triple, considering each triple as a document and each triple element as a term. The predicate statement was not used for indexing since they do not contain relevant information for our study (see Listing 4.1). To keep only significant information and better readability, during indexing we tokenized each triple element by removing part of the URI and special characters like quotes, “>” and “<”. We selected for indexing only the last token from URI in triple statements, as is showed on Listing 4.1. The indexed information using this strategy, containing Wikipedia structures and infobox compositions, was necessary to make a consistent analysis of Wikipedia structured data and also investigate the use of Categories for schema definition.

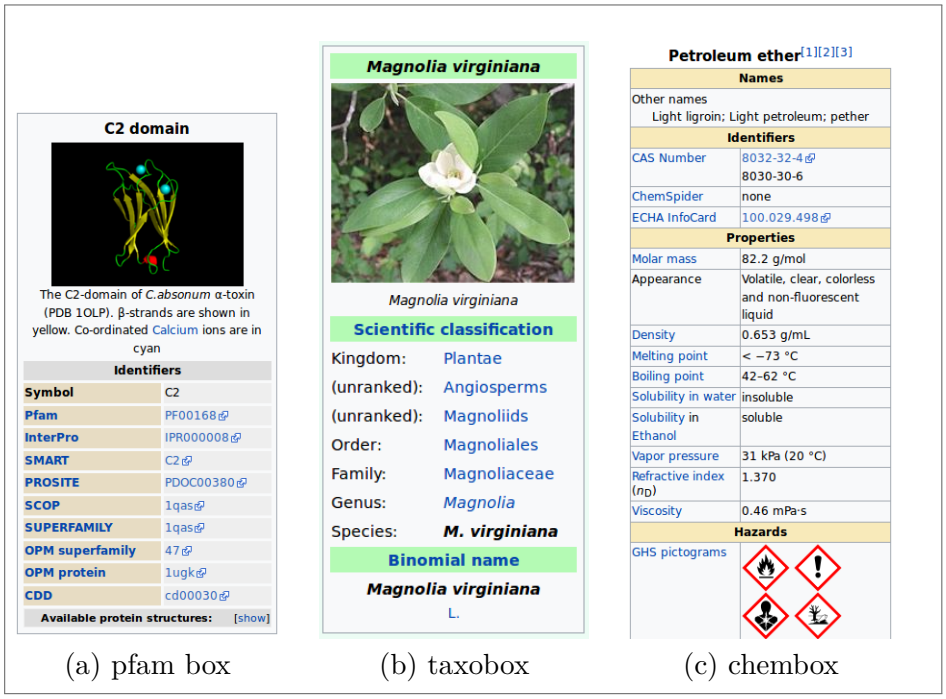
## 4.3 INDEXING WIKIPEDIA INFOBOXES

In addition to Categories and Infobox Templates information, we are also interested in data contained in the Infobox instances. Instead of using DBpedia datasets to retrieve Infobox information, we have parsed the Wikicode in Wikipedia Dump retrieving this information. The decision was made because of the coverage problem of DBpedia, where there are Wikipedia entities yet to be mapped to its ontology, and consequently extracted. We have used the Wikipedia Dump to extract infoboxes, infoboxes template mappings, infobox properties and its respective values preserving original article information, and extending information coverage. The variety of Infobox Templates have also come to the attention. Therefore, we have gathered some known Infobox Types to evaluate its use in the whole Wikipedia.

Lehmann et al. (2015) has pointed out that a wide range of infobox templates are used and, although this system has evolved, some problems can still be highlighted about its use by different communities of Wikipedia editors. Among some problems are the use of

[4] <https://www.w3.org/2009/08/skos-reference/skos.html>

**Source: Wikipedia**




(a) pfam box (b) taxobox (c) chembox

Figure 6 – Infobox types.

different templates to describe the same concept; different templates using different names for the same attribute; and assignment of different formats and units of measurement to attribute values.

Given the problems as mentioned above, as a preliminary step to guide the indexing of Infoboxes from Wikipedia, we have identified different infobox mappings. Beyond the Infobox mapping already shown (Figure 5), there are different Infobox mappings as shown in Figure 6. In spite of the structure be visually the same, the mapping codes change according to template type (*pfam_box*, *taxobox* or *chembox*). Some of the templates are related to specific domains (e.g., *chembox* is associated with the chemicals and *pfam_box* to protein family/domain) whereas the template *Infobox* is adopted across general domains.

Wikipedia presents a statistic page[5] which displays infoboxes statistics compiled in August 2013. Hence, to get recent information we have quantified the infoboxes usage across the Wikipedia DUMP used in this research (dating from October 2016). We have navigated through it counting occurrences of a sample of identified template types. Wikipedia statistics page itself highlights that its only possible know a minimum figure for the number of infoboxes, the true figure will always be higher and less easy to count, once it encompasses a variety of *non-{Infobox}* infoboxes. We have empirically listed some infoboxes types, see Table 1 for the regular expressions used for each one and their proportion of the total number of identified templates.

[5] <https://en.wikipedia.org/wiki/Wikipedia:WikiProject_Infoboxes/Statistics>

Table 1 – Infobox Types Usage Proportion

| **Infobox Pattern** | **Proportion** |
|---|---|
| *{{\s?(I/i)nfobox.*\n(/.*\n)*}}* | $88.49 \cdot 10^{-2}$ |
| *{{\s?(T/t)axobox.*\n(/.*\n)*}}* | $9.48 \cdot 10^{-2}$ |
| *{{\s?(S/s)peciesbox.*\n(/.*\n)*}}* | $52.72 \cdot 10^{-4}$ |
| *{{\s?(G/g)eobox.*\n(/.*\n)*}}* | $52.23 \cdot 10^{-4}$ |
| *{{\s?(C/c)hembox.*\n(/.*\n)*}}* | $37.71 \cdot 10^{-4}$ |
| *{{\s?(A/a)utomatic(_/\s)taxobox.*\n(/.*\n)*}}* | $23.27 \cdot 10^{-4}$ |
| *{{\s?(D/d)rugbox.*\n(/.*\n)*}}* | $22.47 \cdot 10^{-4}$ |
| *{{\s?(E/e)nzyme.*\n(/.*\n)*}}* | $12.04 \cdot 10^{-4}$ |
| *{{\s?(P/p)fam(_/\s)?box.*\n(/.*\n)*}}* | $2.06 \cdot 10^{-4}$ |
| *{{\s?(P/p)rotein.*\n(/.*\n)*}}* | $37.89 \cdot 10^{-6}$ |

**Source:** This research

Templates starting with "Infobox" are the most common ones – 88.4% of the total templates – whereas the other ones cover a tiny proportion of articles. For this reason, in this work, we have considered either for indexing and analysis only the infobox templates mapped with prefix *"Infobox"*.

As mentioned above, we have tried to use DBpedia datasets that contain Infobox data. However, these datasets present some issues related to information relevance and coverage. The **infobox_properties** (52.7M triples) contains raw properties extraction. Contains all information from structured sources in the page. Hence, extracting a lot of noisy data like image frames information throughout article content, e.g., image caption, width, alt or map caption, which are not relevant as an entity attribute. The **infobox_properties_mapped** (27.5M triples) enclose data extracted following DBpedia ontology, but the tuples are in a less clean property namespace, i.e., the object statement of the tuples are wordy, and the predicates of this dataset points to *"rdf:Property"*, a basic denotation of relationship. Lastly, the **mappingbased_literals** (14.4M triples), which follows the ontology namespace, extracting only ontology mapped entities, and as a consequence reducing information coverage.

A problem while using Wikipedia DUMP was the page format, which consists of a syntax used by the MediaWiki software to generate Wiki pages, called Wikicode or Wikitext. A snippet of the Wikicode used to generate infoboxes is shown in Figure 5, note that not all property names used in the Wikicode are kept the same after the page be rendered. To retrieve rendered infobox information we have tried crawl into Articles Hypertext Markup Language (HTML) pages. However, this approach has returned noisy and incomplete Infoboxes information. Hence, the approach applied in this work was to navigate through the dump parsing the Wikicode with *mwparserfromhell* [6], indexing

[6] <https://mwparserfromhell.readthedocs.io/en/latest/>

recovered information using Apache Lucene. We have used the same strategy applied to DBpedia turtle files. We iterate over each tuple from an existing infobox's article and index it in form of a triple: *(<subject> <predicate> <object>)* the article name, property name and property value, respectively. It also indexes the infobox template mapping as a tuple: *(<s> <o>)*: subject (article name) and object (infobox template mapping).

## 4.4 INDEXING WIKIPEDIA TEXTS

To make extractions from Wikipedia articles, it is necessary to retrieve its plain text. Wikipedia is an initiative from the Wikimedia Foundation and makes use of the MediaWiki, a Wiki software designed to display and share open content on the web. Wikis based on MediaWiki usually present an edit button to permit collaborative editing and construction of pages by users without knowledge of HTML or Cascading Style Sheets (CSS). MediaWiki makes use of a simplified markup language, sometimes known as Wikitext, Wikicode or Wiki markup.

As a first step to retrieve plain text from Wikipedia articles, we have tried to parse page's Wikicode with the already used *mwparserfromhell*. This approach was made to recover many texts as possible, as well as when processing infobox information. However, the number of templates, tables, and Uniform Resource Locator (URL) markups presents on these pages have made the process difficult, retrieving the text with noisy data.

The lastly published version of DBpedia, used by this work, includes three large datasets in the NLP Interchange Format (NIF) (HELLMANN et al., 2013), containing the entire text of the wiki-page. These datasets were recently included on DBpedia as an effort to support Computational Linguistics, and NLP tasks and are listed below.

- **nif-context**: contains 26.9M triples, mapping the full text of Wikipedia pages;
- **nif-page-structure**: it has 656.9M triples, mapping the structure of wiki pages, e.g. Section, Paragraph and Title;
- **nif-text-links**: it is composed by 928.3M triples and contains all in-text links of a wiki page.

Proposed by Hellmann et al. (2013), the NIF is an RDF/OWL-based format that aims to achieve interoperability between NLP tools, language resources and annotations. Its ontology[7] contains seven core URIs (String, RFC5147String, Context, isString, referenceContext, beginIndex, endIndex) that provide the foundation to express NLP annotations effectively in RDF. As described by Hellmann et al. (2013) the main class in the ontology is *"nif:String"* which is the class of all words over the alphabet of Unicode characters and it can be used to include the reference text as a literal within the RDF.

[7] <http://persistence.uni-leipzig.org/nlp2rdf/ontologies/nif-core/nif-core.html>

**Source: This research**

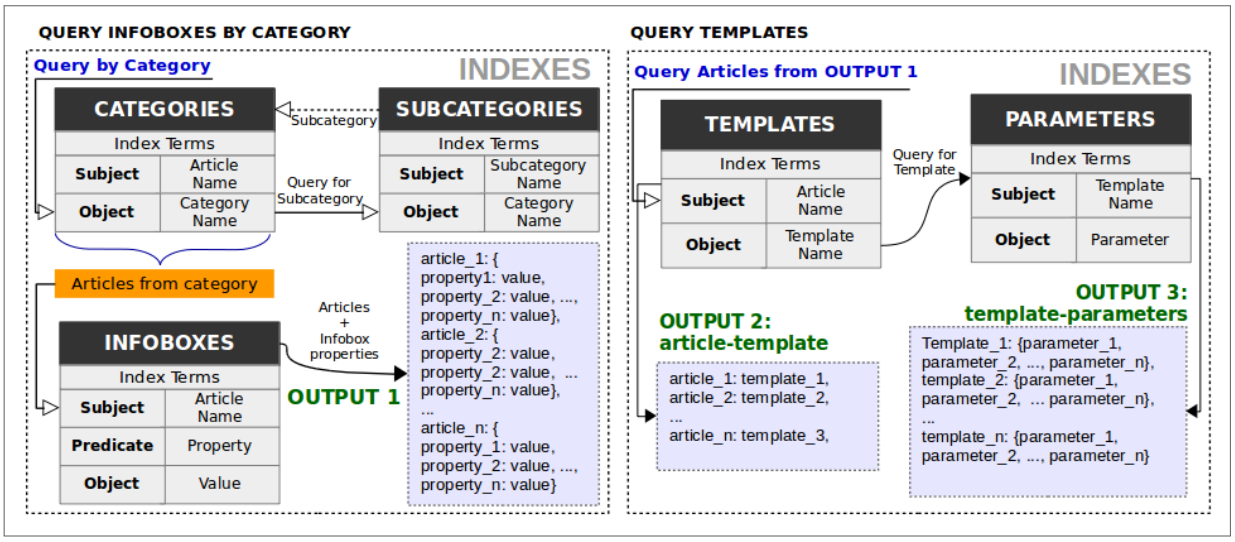



Figure 7 – Framework architecture for querying and retrieving Infoboxes information across Wikipedia categories. LEFT: Query all infoboxes given a Category name. RIGHT: Query mapped infobox templates and correspondent scheme for returned Articles.

Therefore, we decided to use DBpedia **nif-context** dataset to retrieve articles plain text. To do so, we have applied the same Apache Lucene indexing strategy previously explained. RDF tuples were filtered to use only the ones containing the *"nif:String"* statement. Tuples were indexed in the form <*s*> <*o*>. The subject statement was stripped to store only resource last token, which corresponds to the entity name. The object statement, representing full resource text, was kept unchangeable. The predicate statement corresponding to *"nif:String"* annotation is ignored.

## 4.5 FRAMEWORK ARCHITECTURE FOR QUERYING CATEGORY INFORMATION

To better understand Wikipedia data distribution we have done an extensive analysis of Wikipedia structured information, i.e., Categories, Subcategories, Infoboxes, and Infobox Templates. Therefore we used DBpedia datasets to capture Wikipedia's categories, subcategories, template schemas, mapped articles to categories and Article's plain text. Also, we have navigated through the Wikipedia dump to query infoboxes information. We tried to use infoboxes information mapped on DBpedia datasets. However, there are some Wikipedia entities yet to be mapped to DBpedia ontology thus being absent on its datasets.

To assist the analysis we have developed a framework using Apache Lucene, where all DBpedia structure information and Infoboxes on Wikipedia dump were indexed and queried. We have used the querying framework to retrieve diverse kinds of information, e.g., top used infobox templates, top used infobox properties, all Wikipedia infoboxes,

categories and templates, as well as all infoboxes, and templates related to a given category.

Querying data related to Category was necessary to investigate schema definition according to different information domains. As an example, the querying strategy shown in Figure 7 is responsible for retrieving all infoboxes, infobox-template mappings and template schemas related to a given Category. The data recovered by this search was used to perform analysis of homogeneity and consistency of infobox schemas under different Categories. The analysis was expected to auxiliary the schema definition for the slot filling task.

**Source: This research**

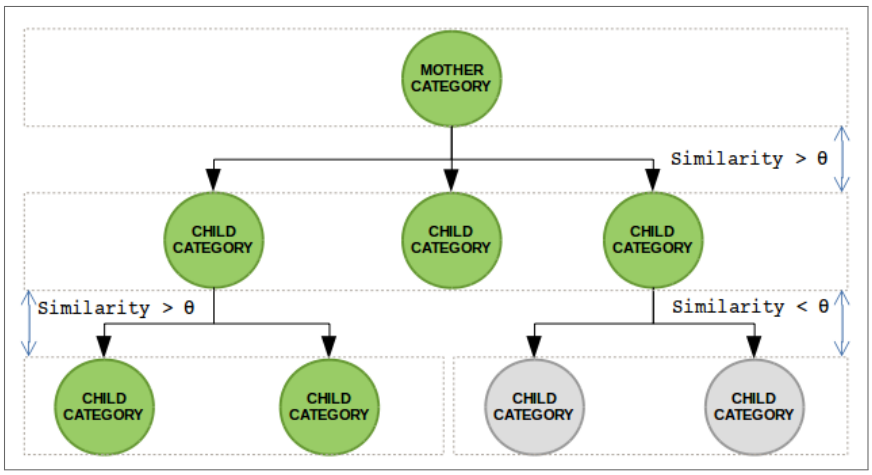



Figure 8 – Breadth-first search to gather infoboxes until given threshold similarity between levels of subcategories

This query receives as input a Category $C$ and returns three outputs: i) the Infobox Instance $I$ for each Article $A$ existing under informed Category $C$; ii) the Infobox Template Mapping $T$ for each returned Article $A$ with an Infobox instance $I$; iii) the Template Scheme $S$ for each Infobox Template Mapping $T$. A breadth-first search is used to navigate across subcategories retrieving registered articles under them, until an informed similarity threshold (Figure 8). For each level in the hierarchy category, the algorithm unifies infobox attributes of found infoboxes, creating some global schema for that level, measuring the similarity between this schema and the ones found on child categories. The search stops when the similarity between child and mother node is below a given threshold $\theta$ (given the heterogeneity of property names and infobox variation, for this study we have defined $\theta = 0.15$). We have used Jaccard to measure the similarity between the common scheme of root level and the next ones. The data analysis of this search result is shown on Chapter 5.

# 5 DATA ANALYSIS

The main objectives of this data analysis are divided into two classes: i) General Statistics: give an overview of structured information on Wikipedia as a whole, showing raw numbers of Articles, Categories, Subcategories, Properties, Templates, and Infoboxes, as well as identify infoboxes inconsistencies; ii) Category-based Statistics: discusses data related to Categories and investigate the identification of common infobox schemes, with attributes from different infoboxes in the same category, trying to define a measure of quality and homogeneity of infoboxes inside a given category.

We hope the findings and analysis on this chapter will help further completeness and augmentation of knowledge bases as DBPedia and YAGO, as well as guide researchers working with scheme discovery and information extraction on Wikipedia.

## 5.1 GENERAL STATISTICS

This section will briefly present some statistics about queried information from developed indexing and searching framework. It starts by showing raw numbers from Wikipedia elements as Articles/Entities, Infoboxes, Categories, Subcategories, Properties and Infobox templates (see Table 2). These data provide information about infoboxes distribution across articles and categories. It also gives a notion about the diversity of templates and properties currently used by existing infoboxes. It is important to emphasize that only a bit more than 50% of existing articles present an Infobox instance. Although not every existing article must have an infobox instance, this proportion shows us the gap of entities yet to be structured.

Table 2 – Overview data of Wikipedia exploration

| **Element** | **Total** |
|---|---|
| *Articles* | 5.166.304 |
| *Infoboxes* | 2.785.031 |
| *Infoboxes w/ geo* | 2.105.172 |
| *Infoboxes w/ datetime* | 1.584.340 |
| *Categories* | 1.079.614 |
| *Properties* | 56.819 |
| *Infoboxes Templates* | 3.448 |

**Source:** This research

Figure 9 shows the distribution of the size of infoboxes on complete Wikipedia. The infobox size is measured by counting the number of distinct properties. The numbers show

**Source: This research**

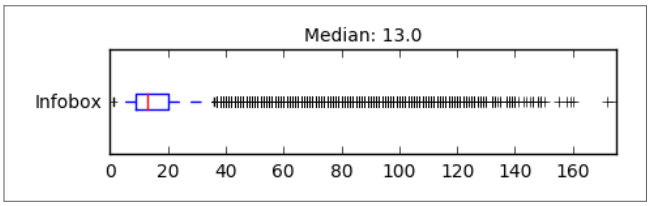



Figure 9 – Distribution of Infoboxes size across Wikipedia.

that there is a reasonable number of properties per infobox: the median size of infoboxes is 13. 68.26% of infoboxes lies within one standard deviation of the mean ($16.24 \pm 10.57$). However, the infoboxes size do not follow a normal distribution. The largest infobox, with 172 properties, corresponds to the *Al Jazeera English* entity, a Qatari paid television news channel, the amount of properties is due to the descriptions of the variation in satellite, cable and IPTV information shown in the Infobox.

**Source: This research**

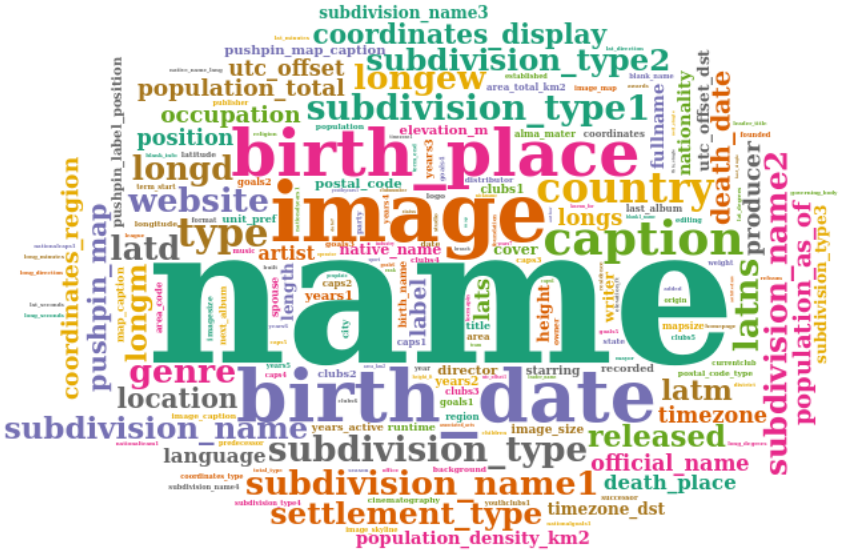



Figure 10 – Top property names used on Wikipedia Infoboxes

Figure 10 presents a word cloud with the most popular properties in the infoboxes. As one can see, some of the most popular properties are from generic entities, for instance, *birth_place* and *birth_date* are attributes of entities related to *Person*. Properties like *name* and *image* can be applied to any type of topic, is not a coincidence that almost every infobox instance comes with a name and an image example. Another observation is that attributes related to geo and temporal information are also very common: e.g., *location*, *timezone*, *country*, *birth_place*, *birth_date*, *death_date*, *years*, and variations of *longitude*, *latitude* and *coordinates*, among others. Attributes that indicate multivalued information are also present, e.g: *subdivision_type*, *subdivision_type1*, *subdivision_type2* and are usually followed by *subdivision_name*, *subdivision_name1*, *subdivision_name2*;

they are mostly related to *Infobox settlement* and are used to organize settlements subdivision types like country, state, seat or province, and its respective values. This information is an indicative that not all attribute names are self-descriptive. Also, it is important to point out that recovered properties where normalized to get its frequencies.

Table 3 – Top 10 Infobox Templates by Usage Proportion

| **Infobox Template** | **Proportion** |
|---|---|
| infobox_settlement | $1.32 \cdot 10^{-1}$ |
| infobox_person | $6.44 \cdot 10^{-2}$ |
| infobox_football_biography | $4.84 \cdot 10^{-2}$ |
| infobox_album | $4.61 \cdot 10^{-2}$ |
| infobox_film | $3.97 \cdot 10^{-2}$ |
| infobox_musical_artist | $2.96 \cdot 10^{-2}$ |
| infobox_company | $1.99 \cdot 10^{-2}$ |
| infobox_nrhp | $1.97 \cdot 10^{-2}$ |
| infobox_single | $1.79 \cdot 10^{-2}$ |
| infobox_officeholder | $1.72 \cdot 10^{-2}$ |

**Source:** This research

Table 3 presents the proportion of the top 10 templates that use the "*Infobox **" naming pattern. Most of them refer to high-level entities. For instance, the top template, *Infobox settlement*[1], is recommended to be used for "any subdivision below the level of a country" (e.g., cities, towns, villages, communities, administrative districts, counties, provinces, among others). The *Infobox officeholder*[2] also incorporates a bunch of other templates (e.g., Infobox president, Infobox governor, Infobox senator, Infobox state representative, Infobox congressman, among others). This reflects the granularity of infobox templates usage and freedom of creation/editing by each Article's editor. Also related to location, *Infobox NRHP*[3] is used on articles about historic places in the USA listed on the National Register of Historic Places.

Other interesting observations from the numbers in Table 3 are: four of the top templates are related to People (i.e., person, football biography, musical artist and office holder); three templates are associated with Arts and Entertainment (i.e., film, album and single); two templates are related to geographic Locations or Places (i.e., settlement and NRHP); and the seventh (7th) top template is related to Organization. This might reflect that they are popular topics on Wikipedia and should provide the right amount of data for training of machine learning or deep learning tasks.

[1] <https://en.wikipedia.org/wiki/Template:Infobox_settlement>
[2] <https://en.wikipedia.org/wiki/Template:Infobox_officeholder>
[3] <https://en.wikipedia.org/wiki/Template:Infobox_NRHP>

## 5.2 CATEGORY-BASED STATISTICS

Table 4 – Extracted categories, number of articles found and subcategory extraction nodes

| Category | Art. | Inf. | Nodes | Category Description |
|---|---|---|---|---|
| *French films* | 6587 | 6249 | 583 | All french films. |
| *Oil companies of the United States* | 464 | 288 | 20 | Contains companies from United States Oil industry |
| *Operas* | 2626 | 402 | 7 | Contains operas, subcategorized by composer, genre, original language, year and acts. |
| *Protein families* | 742 | 379 | 1 | Collects together articles describing sets of related proteins called a family. |
| *Skyscrapers between 100 and 149 meters* | 1028 | 905 | 1 | Tall buildings between 100 and 149 meters |
| *HarperCollins books* | 1832 | 1597 | 15 | Books published by HarperCollins and its imprints - a subsidiary of News Corp, based in New York City. |
| *Towns in Turkey* | 750 | 693 | 7 | Turkish cities |
| *Alpine three-thousanders* | 944 | 934 | 2 | Mountains between 3,000 metres (9,842 ft) and 3,999 metres (13,122 ft) above sea level in the Alps, in Austria, Switzerland, Italy and France. |
| *Numbered minor planets* | 2959 | 2910 | 1 | All articles on numbered minor planets (MPs). |
| *IOS games* | 1799 | 1730 | 6 | Category for video games available for the iOS operating system. |

**Source:** This research

In this section, it is investigated the usage of Categories to relate entities semantically, different from Nguyen et al. (2010) which link entities through references over its content, and Nguyen et al. (2012) that presents an Infobox clustering strategy to discover infobox types. The strategy discussed here tries to assist the schema discovery process through data analysis and proposed quality measures. Thus, this analysis will investigate the inferring of data schema from the Categories hierarchy. Based on Categories hierarchy this approach can benefit schema definition in corpus organized as hierarchies. Hence, we look at the structured information of Wikipedia on a category level.

The Oxford Online dictionary defines Category as *"a class or division of people or things regarded as having particular shared characteristics."*, based on this assumption our goal is to study how the structured information is organized in different categories and try use information under a given domain for automatic schema discovery.

Since our Wikipedia dump has more than 1 million categories, we selected ten categories to perform this analysis. As among the top-10 categories there were many categories on the same topic and related with general concepts (e.g., films, company, settlement), we have opted to manually select those which corresponds to different domains and more

specific concepts (e.g., French films, oil companies of the United States, towns in Turkey). To choose them, we ranked all Wikipedia categories based on the number of articles with "*Infobox *something**" instances on its root hierarchy level, i.e., subcategories were not considered. We have manually chosen Categories inside the top-200 ranking of infobox instances on the root level. Table 4 shows the categories used in this study, for each category are shown the number of articles, the number of infoboxes and a brief description of each category domain. Retrieved data about these categories were acquired using the searching framework scheme described on Section 4.5, where subcategories were explored to gather more data and specificities about the domain. In the table are also shown the number of matched subcategory nodes in the hierarchy.

**Source: This research**

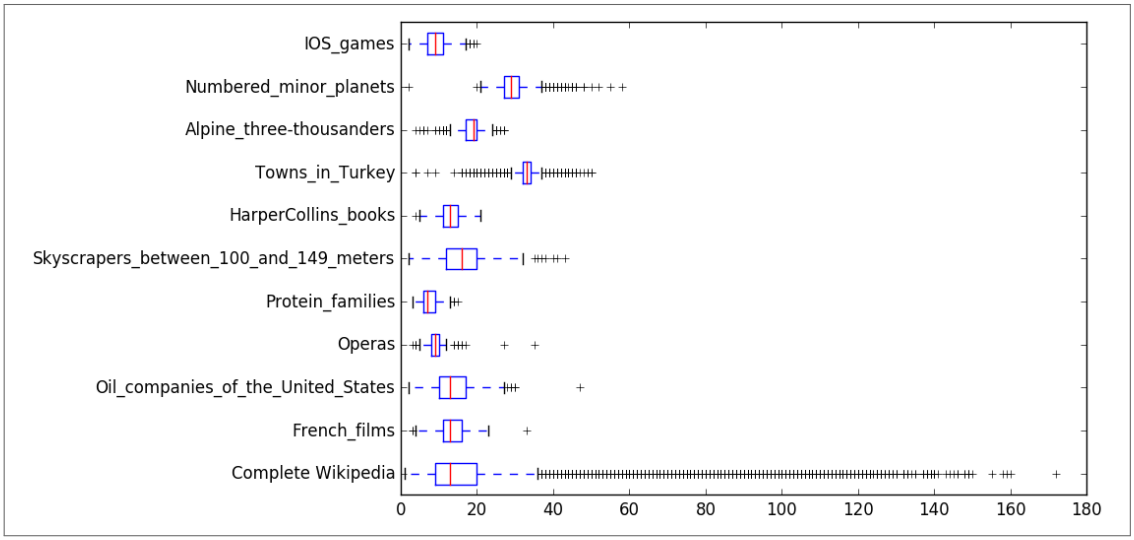



Figure 11 – Distribution of infoboxes size by category

Figure 11 shows the distribution of the size of the infoboxes for the ten categories and the complete Wikipedia. The median size of infoboxes categories is between 7 and 20, except for *Numbered Minor Planets* which is 30. The overall median size of infoboxes in the datasets is around 13, which is close to the median for all infoboxes as presented previously. The category with the largest median schema size is *Numbered Minor Planets*. Its specificity and scientific nature can explain this. The common attributes in its infoboxes are properties for orbital and physical characteristics as well as proper orbital elements of planets, which suggests that this category is curated by a community of specialized and accurate users. Figure 11 also shows that there is not much variation in size on each category. The most considerable variation is *Skyscrapers between 100 and 149 meters* with an interquartile range of 8, followed by *Oil companies of the United States* with 7.

Table 5 – Overview of templates usage for each category. Total count of used templates, most used template, template size and proportion of the most used template

| Category | Templates | Most Used | Size | Prop. |
|---|---|---|---|---|
| *French films* | 11 | Infobox film | 36 | 0.9916 |
| *Oil Companies of the United States* | 29 | Infobox company | 84 | 0.6458 |
| *Operas* | 10 | Infobox opera | 25 | 0.9179 |
| *Protein Families* | 8 | Infobox protein family | 21 | 0.9366 |
| *Skyscrapers between 100 and 149 meters* | 12 | Infobox building | 182 | 0.9259 |
| *HarperCollins books* | 15 | Infobox book | 66 | 0.9530 |
| *Towns in Turkey* | 6 | Infobox settlement | 424 | 0.9870 |
| *Alpine Three-thousanders* | 4 | Infobox mountain | 239 | 0.9914 |
| *Numbered minor planets* | 2 | Infobox planet | 119 | 0.9996 |
| *IOS games* | 17 | Infobox video game | 52 | 0.8924 |

**Source:** This research

We also studied how the categories use templates. Table 5 presents some numbers regarding that. Although there is some variation regarding the number of templates used by the categories, almost all of them use primarily a single template for the articles: the most used template in analyzed categories covers about 90% of the articles. The only exception is the category Oil companies of the United States whose the most used template covers only 64% of the articles. It can indicate that this category is more heterogeneous than the other ones but wrong mappings are also the problem. Some articles of this category use the template infobox_oil_field whereas others infobox_oilfield, which have the same properties. Another issue in this category is wrong mappings: mappings for *Infobox_NASCAR_driver*, *Infobox_racing_car* and *Infobox_court_case* templates are under this category. It occurs because those infobox mappings are related to subcategories as Chevron Corporation, a subcategory of Oil Companies of the United States. This behavior can indicate that for this category the applied threshold for searching was excessively relaxed.

Some templates present a large schema (e.g., Infobox Settlement and Infobox mountain with 424 and 239 properties, respectively). However, Infobox templates are flexible, and its suggested properties are not mandatory.

### 5.2.1 Categories Quality Analysis

We have conceptualized two quality indexes to analyze the composition of infobox schemata from Wikipedia. These measures are calculated considering queried information from Categories (as shown previously). They are divided according to its type of input data: the external scheme-based quality index uses data about suggested template schemes (external from category) and the recovered Infoboxes schemes; the internal scheme-based index uses only data about the recovered infoboxes schemes (internal from category).

The external scheme-based quality index can help define the trustworthiness of infoboxes on a given Category to follow suggested template schemes, i.e., the coverage of infoboxes attributes of mapped templates. Internal scheme-based quality indexes can help define a homogeneous infobox scheme for Category based on the similarity between existing infoboxes.

#### 5.2.1.1 What is the coverage of infoboxes' attributes from mapped templates?

As we mentioned before, infobox templates can help to create more homogeneous infoboxes. For that, it is necessary that the created Infobox apply the attributes suggested by the mapped template. We define, therefore, as a quality measure the coverage of infoboxes' attributes used from their mapped templates, which we call template coverage. This measure can be obtained through the following proposition:

$$g(\delta, \gamma) = \frac{\|\vec{\delta} \cap \vec{\gamma}\|}{\|\vec{\gamma}\|} \tag{5.1}$$

$$\tilde{\kappa} = \text{median}_{(\delta,\gamma)\in C}\{g(\vec{\delta}, \vec{\gamma})\} \tag{5.2}$$

Where:
$\delta$, infobox instance in category
$\gamma$, template mapped to $\delta$ infobox instance;
$\vec{\delta}$, properties actually used on infobox definition;
$\vec{\gamma}$, suggested properties from mapped template; and
$\tilde{\kappa}$, general coverage index for Category $C$.

**Source: This research**

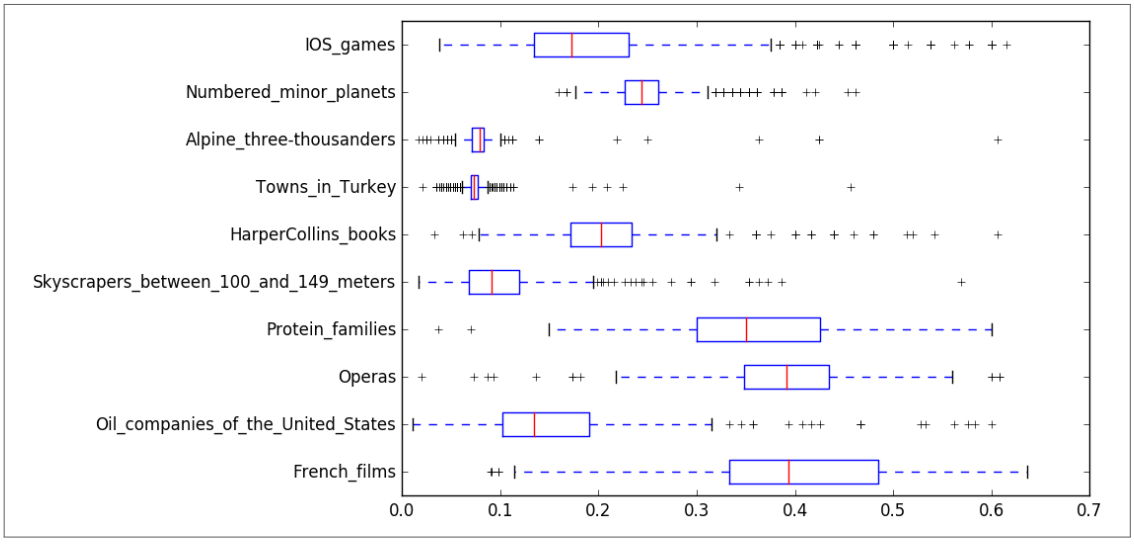



Figure 12 – External scheme-based quality index for categories. Properties from suggested templates' schema used on internal schemata

To get the intersection between suggested template schemata and infobox schemata we have normalized all property names through lowercasing and removing underlines, dashes, spaces, and duplicates. The result of this analysis for selected categories can be seen in Figure 12. The numbers show that there is a great variation in the median of the intersection for analyzed categories. Whereas the categories French Films and Operas have a value closer to 0.4, Towns in Turkey it is close to 0.1.

**Source: This research**

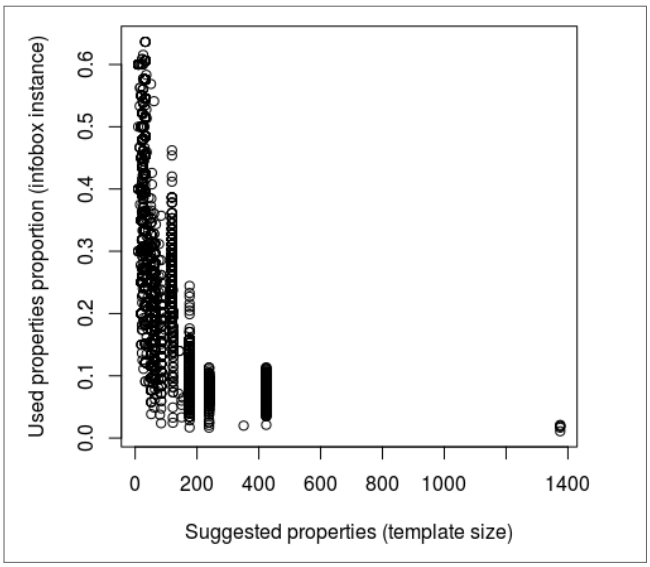



Figure 13 – Correlation between Suggested properties from Infobox templates and the proportion of properties actually used on Infobox instance

Comparing those values with the size of the largest suggested template for each category presented in Table 5, it seems there is some negative correlation between them, indicating that the template size may influence the quality index. To verify this observation, we have calculated for each category the proportion of used properties and the size of the suggested template. We calculated for each category the Spearman correlation between template coverage and mapped templates size: $\rho$=−0.779669. Under a significance level of 0.95, the test presents $p$-value $< 2.2e^{-16}$, which indicates a statistically high significance between these variables. To illustrate it, a scatter plot between these two features can be seen on Figure 13. As one can see, there is a template with almost 1400 properties, which corresponds to *Infobox officeholder* template on category *Oil Companies in the United States* that presents around 2% of suggested properties used in infobox instances.

#### 5.2.1.2 How similar are the infoboxes on a category?

In this section, we evaluate the homogeneity of the attributes of the infoboxes belonging to the same category. One can assume that the more similar the infoboxes' attributes are on a category, the closer is to define a single schema for it.

The intuition behind this measure is while Categories can present diverse infobox types (as shown in Table 5), their schemes can be very similar. Therefore, instead of handle a vast number of infobox types under the same domain (Category) we could define only a common infobox scheme for each Category. We have normalized recovered properties, by removing underlines and lowercasing them. We have used Jaccard similarity to measure the similarity between infobox attributes queried from the same category. The measure can be obtained through the following proposition:

$$f(\Delta) = jaccard(\vec{\delta_i}, \vec{\delta_x}), \{\forall \delta_i \delta_x \in \Delta \mid i = 1..(n-1), x = (i+1)..n\} \tag{5.3}$$

$$\tilde{\chi} = \text{median}\{f(\Delta)\}, \forall \Delta \in C \tag{5.4}$$

Where:

$\Delta$, matrix of infobox instances, each column corresponds to a property name;

$f(\Delta)$, condensed similarity matrix; and

$\tilde{\chi}$, general similarity index for Category $C$.

In Figure 14, we present boxplots with the distribution of the similarity for the ten categories. The plots show that in general there is not much homogeneity between infoboxes' schemas. The most heterogeneous categories are *Numbered minor planets* and *Alpine three-thousanders* with the median around 25%. Although category *Numbered minor planets* is composed by only two infobox types (template mappings), where *Infobox planet* has a frequency of 99%, it presents 119 suggested properties, and in average only 25 properties are used on infobox instances for this category. Nonetheless, it is not guaranteed that used properties are always the same. See the low properties coverage from suggested templates shown in Figure 12 for this category. The two plots also infer a negative correlation between external schema-based (Figure 12) and internal schema-based (Figure 14) indexes, where categories presenting a high index of suggested properties usage usually presents a low index of schema homogeneity and vice versa, i.e., the greater the properties coverage, less the homogeneity. This behavior is consistent, once the higher use of suggested properties indicates the presence of specific information for the infobox type.

The most homogeneous category is *Oil companies of the United States* with median around 90%, even though being composed of 24 templates and the most used template (Infobox company) shows a frequency of only 62%, the median of the external schema-based index is around 15%. This can be explained by suggested scheme size and the existence of shared properties across different suggested templates, e.g. Infobox Company, Infobox Building and Infobox NRHP, the top 3 infobox templates most used by the category *Oil Companies of the United States*, share properties like location, name, owner, type, architect, engineer and other small naming variations.

**Source: This research**

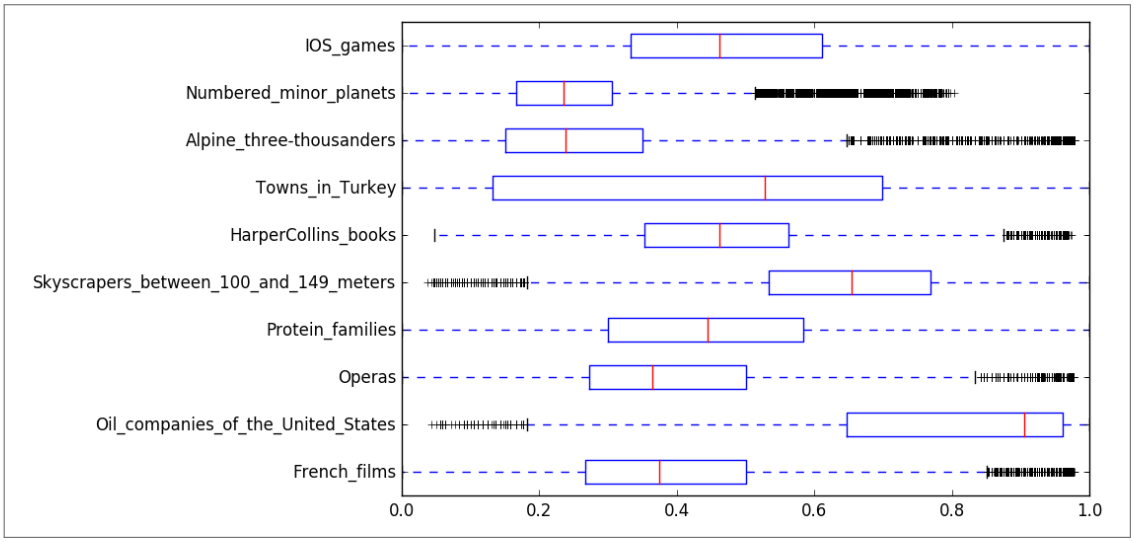



Figure 14 – Internal scheme-based quality index for categories. Schemata homogeneity obtained through Jaccard similarity between infobox schemas.

We can conclude from these numbers that: (1) there is a great heterogeneity of infoboxes within a same category; (2) the number of templates used by the infoboxes in a category has not much influence in its heterogeneity; and (3) the size of the suggested template may increase the heterogeneity, once not all properties must be used and different properties are applied to only some entities under the same domain (subtypes).

#### 5.2.1.3 Would it be possible use Wikipedia categories to define common scheme for entities under the same domain?

Wikipedia Categories should aggregate information under the same topic, and consequently the same domain. Hence, it is possible to gather some similarities in infobox schemes under them. The main questions to be answered are: (1) How define a limit of visited subcategories maintaining infobox similarities? (2) Infobox instances under the same category hierarchy and mapped to different templates, represent a same entity type?

We expect to have cleared some questions and raised new ones through the shown data analysis. But a lot of work yet must be done to acquire an automatic schema discovery on Wikipedia infoboxes through a Category approach. We believe the starting point is deciding whether the discovery process is from a general or more specific schema.

# 6 DEEPEX

This chapter presents Deepex: the solution to the underlined problem of Knowledge Base Completeness through information extraction on Wikipedia.

**Source: This research**

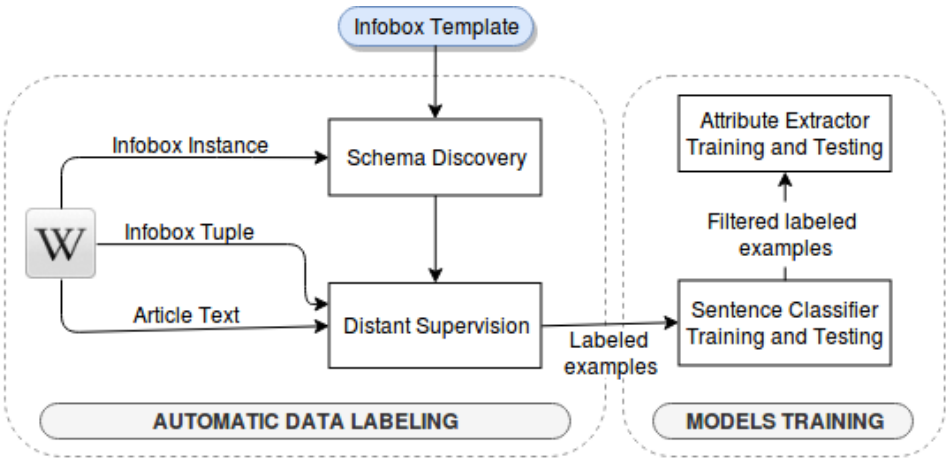



Figure 15 – Deepex architecture for building datasets, classifiers and extractors

The architecture of our system is presented in Figures 15 and 16. The architecture showed in Figure 15 is composed by the two modules described bellow:

- **Automatic Data Labeling** This module presents two main components. The first one, *Schema Discovery* receives as input the name of the infobox template that represents the information domain to be structured. It collects infobox instances that use the given template and computes the most used properties across them, which we call the template's schema. The second component, the *Distant Supervision*, automatically labels the data. It receives as input the previously defined schema from the *Schema Discovery* component. The *Distant Supervision* component also queries infobox tuples from infobox instances, using the given template type. For each retrieved infobox instance, it also retrieves the respective article text. To build the training datasets it applies DS between the retrieved tuples and articles sentences. The training samples are composed of two types of labels, one at sentence level to train sentence classifiers and the other at token level to train extractors. The output of this component is a set of labeled datasets, each dataset corresponding to a property in the defined schema;

- **Models Training** comprises the training of the sentence classifiers and attribute extractors. The *Sentence Classifier Training* receives as input the labeled sentences coming from the *Distant Supervision* component. It trains the sentence classifiers

that will be used to predict informed sentences may or may not contain a value for the respective property, i.e., the component is used to filter sentences before passing them to the attribute extractor. The *Attribute Extractor Training* uses the labeled tokens that comes from *Distant Supervision* component. However, before starting building the extractors, the samples are filtered by the *Sentence Classifier Training*;

**Source: This research**

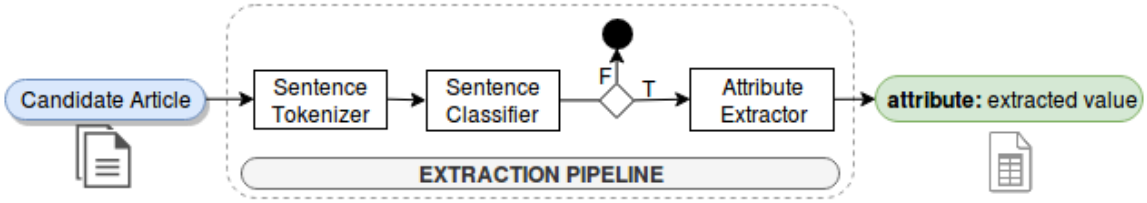



Figure 16 – Deepex pipeline extraction architecture

Figure 16 presents the pipeline extraction architecture executed after building datasets, classifiers and attribute extractors. It is described below:

- **Extraction pipeline** consists in processing candidate articles to extract attributes in the domain. A *Sentence Tokenizer* is applied to the text of the candidate article. The processed sentences are passed through the *Sentence Classifier* which filters sentences that may contain values related to the respective property and send them to the *Attribute Extractor* which performs the attribute value extraction.

Details of each module from the architecture are provided in the following sections. Section 6.1 presents the target corpus and supervision knowledge base definition, as well as the training datasets creation using the DS assumption aligned with a similarity measure. Section 6.2 shows the modeling and training of sentence classifiers and extractors.

## 6.1 AUTOMATIC DATA LABELING

This module presents two main components, the *Schema Discovery* and the *Distant Supervision*. The first component defines the domain schema, i.e., the set of properties of the domain. The domain schema is used by the processes of training classifiers, extractors, and the extraction itself. The second component, *Distant Supervision*, generates labeled data for each property of the schema through the matching of tuples and sentence tokens.

### 6.1.1 Schema Discovery

Our Extraction task requires a previously defined schema to be filled. Our first attempt to determine the domain schema was through the analysis of Wikipedia Categories and

subcategories, to conceptualize domains and infer common properties under them. However, as presented in Chapter 3 traditional approaches using Wikipedia for slot-filling tasks have relied on the most common properties from Template types. Hence, we have followed the same path.

To make easier processing, retrieving and searching for information on Wikipedia it was required the development of storage and querying strategy that scales to the size of its corpus without requiring high computational effort. For that we have used Apache Lucene, the strategy is presented in Chapter 4.

The component receives as input the Infobox Template name representing the information domain for extraction. A query is done to the indexed Wikipedia Corpus to retrieve all Infobox instances using the given Template name. All Infobox properties are counted and sorted. Following the work of Wu and Weld (2007) a threshold of 60% of frequency was defined. All properties with a frequency of use above this threshold are considered as part of the schema.

### 6.1.2 Distant Supervision

Inspired by the works of Wu and Weld (2007), and Lange, Böhm and Naumann (2010) we have applied Distant Supervision as a paradigm for automatic labeling of training examples. Using Wikipedia corpus and infobox instances as the source of supervision we expect to align entities and its properties to text sentences successfully. This strategy provides training data for sentences classification and entities extraction.

The component receives as input the tuples from infobox instances using the previously informed infobox template, and the respective article text containing this infobox instance. It also receives from *Schema Discovery* the set of properties composing the schema for the domain, which will lead the automatic dataset labeling. The output is a set of labeled datasets, one dataset for each property in the schema. Each dataset contains labeled sentences that are sent for *Sentence Classifier Training*, and labeled tokens that are used for *Attribute Extractor Training*.

As previously explained in Section 4.3 we have tried to apply DBpedia as the source of entities' attributes to perform distant supervision. However, DBpedia shows a lack of coverage of Wikipedia entities. To encompass more entities on different domain contexts we decided to follow the same approach of Wu and Weld (2007), and Lange, Böhm and Naumann (2010). These works have used infobox tuples from Wikipedia Dump as sources of supervision for labeling tokens in the sentence using Distant Supervision matching. While using Wikipedia Infoboxes have allowed gathering a variety of entities, it has also aroused new challenges related with information retrieval which we have worked around through the processing described on Section 4.3.

Our distant supervision strategy presented in Figure 17 uses the tuples from infoboxes instances as sources of supervision to label attributes present in the sentences of the

Listing 6.1 – Wikicode creating Infobox instance for entity Caribou County.

```
{{Infobox U.S. county
| county              = Caribou County
| state               = Idaho
| founded year        = 1919
| founded date        = February 11
| seat wl             = Soda Springs
| largest city wl     = Soda Springs
| area_total_sq_mi    = 1799
| area_land_sq_mi     = 1764
| area_water_sq_mi    = 34
| area percentage     = 1.9%
| census estimate yr = 2017
| pop                 = 7034
| density_sq_mi       = 3.9
| district            = 2nd
}}
```

**Source: Wikipedia**

articles. A simple strategy to perform this task would be to do exact matching between tokens of the values and the properties with sentences tokens. As explained by Takamatsu, Sato and Nakagawa (2012), Riedel, Yao and McCallum (2010), Hoffmann et al. (2011), and Surdeanu et al. (2010) this assumption tends to introduce some noise, especially, when the source of supervision does not come from the same target corpus. Although Wikipedia Article's text and infoboxes present the same scope, usually some problems are found when dealing with properties names and values spelling.

An infobox tuple is composed by a property name and its respective value(s). It was built one dataset for each property existing on the given Infobox schema (properties of the domain). Because the property names come from the Wikicode vocabulary, it is even more difficult to do exact matching with plain text sentences. In Listing 6.1 are presented some infobox tuples from an entity mapped to *Infobox U.S. county* type. Beyond underlines and joint words, there are property names ending with "wl" as well as abbreviations, e.g, pop, sq, mi, and yr (for more variation in spelling of property names, see Figure 10).

To lead an autonomous DS matching without generating too much noise or loss of information coverage, we decided to apply the Soft TD-IDF (with Jaro Winkler)(COHEN; RAVIKUMAR; FIENBERG, 2003) similarity measure to match between property-value tokens and sentence tokens. See Figure 17, where we have a snippet of a Wikipedia Article, and an example of infobox tuple. The sentence example is being passed over by a sliding window, which tries to match with Soft TF-IDF the tokens within the window with each other token from tuples property and value. The sliding window greatest score according to a given threshold indicates the sentence label, while the individual Soft TF-IDF score gives the NER tagging. Any other similarity measure could be used for this task, e.g. Jaccard Similarity. However, instead of measuring the intersection over union of word characters, like Jaccard Similarity would do, the Soft TF-IDF will give higher scores to common words in the corpus and words sharing the same prefix.

The Soft TF-IDF was first proposed by Cohen, Ravikumar and Fienberg (2003) and consists of a variation of the TF-IDF weighting. Its calculation (DOAN; HALEVY; IVES, 2012) is obtained through proposition (6.1) . Different from the basic approach, this metric makes use of the Jaro Winkler (WINKLER, 1999) measure to define the similarity level between words. Furthermore, the Jaro Winkler applies the Jaro (JARO, 1989) measure aligned with the prioritization of strings sharing the same prefix. This approach was applied to label sentences as having high, low, or borderline confidence in presenting property value(s). Also, to make NER tagging at the tokens level to further training of extractors (see Figure 17).

$$s(x,y) = \sum_{t \in close(x,y,k)} v_x(t) \cdot v_y(u_*) \cdot s'(t,u_*) \tag{6.1}$$

Where:

$x$ and $y$, two strings

$k$, a prespecified threshold (Default value is 0.5. We have considered a high threshold of 0.8, to ensure a minimal difference between the tokens)

$s'$, is a basic similarity measure (e.g., Jaro Winkler)

$close(x,y,k)$, set of all terms $t \in B_x$ that have at least one close term $u \in B_y$, that satisfies $s1(t,u) \geq k$. $B_x$ is the document of $x$, and $B_y$ is the document of $y$;

$v_x$ and $v_y$, the TF/IDF feature vectors normalized to length 1.

We consider a match when the Soft TF-IDF score between the tokens in the sliding window and the tokens of the property and values tokens is higher than 0.5. If multiple

**Source: This research**

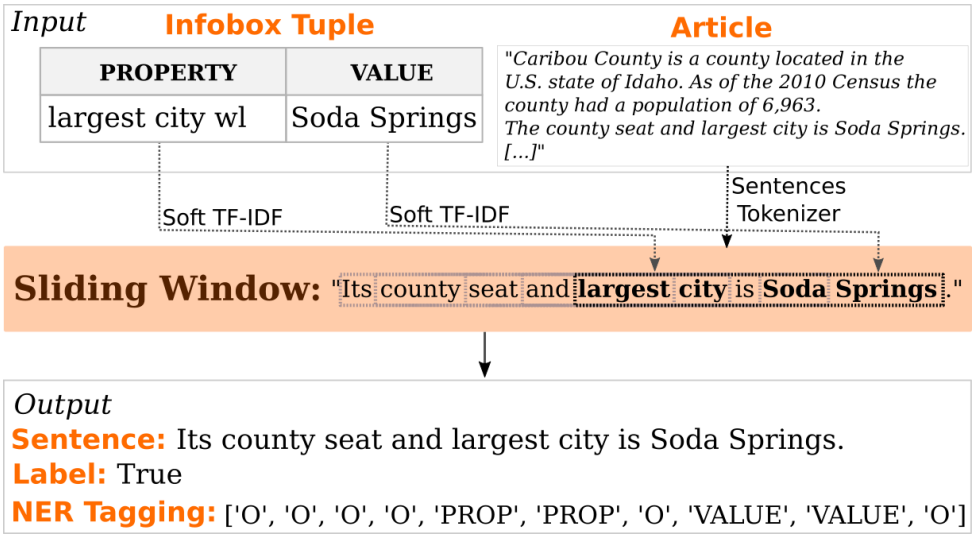



Figure 17 – Example of sliding window for labeling sentences and NER tagging using Soft TF-IDF (with Jaro Winkler) measure.

sentences have a score higher than 0.5 for the given property-value, the solution picks the sentence with the highest score.

We have empirically defined a sliding window of size 5 (five) tokens. When the sliding window presents a similarity score above or equal to 0.5 it indicates that the sentence is a relevant example, i.e., "positive" examples. The sentences with the greatest sliding window expressing any score of similarity between 0.0 (zero) and 0.5, were considered as borderline samples, thus being labeled as "other" — lastly, the sentences were all the sliding windows have showed similarity equal to 0.0 (zero) where considered as having low confidence and were labeled as "negative" training examples.

We have defined the labeling of borderline sentences to gather unusual sentence patterns. There are cases where in spite the fact that the property-value relation exists, the sliding window matching presents low confidence. For instance, this is the case of long sentences or different spellings between the tuple and the sentence. An example of sliding window for long sentence that would present low confidence at first sight is the matching for the tuple (*area_land_sq_mi: 1,764*) and the sentence *"...the county has a total area of 1,799... and 34 square miles (88 km$^2$) (1.9%) is water..."*. The gap between the matching words *area* and *water* to *34 square miles* is bigger than 5 words. This may cause a low similarity confidence to label the sentence as positive on the first run.

Given the exposed, the sentences labeled as "other" are passed through the sentence classifier. Depending on the new confidence degree, these sentences will have their labels updated to "positive" or "negative". Positive and negative samples are used to train sentence classifiers, while only positive samples are used to train extractors.

To label the tokens for training the extractor, we have applied the Soft TF-IDF (with Jaro Winkler) on a token by token basis, keeping the matched pair with the biggest score, separately comparing property and value tokens with the ones belonging to the sentence. Matched tokens from property name were tagged as "PROP", while matched tokens from value were tagged as "VALUE". Remaining tokens with no matching were tagged as "O" (from "other"). The example for the NER tagging system is also shown on Figure 17

## 6.2 MODELS TRAINING

The distant supervision provides labeled examples to build the sentence classifier and attribute extractor. The output of the components are the trained models to be used by the Extraction Pipeline module.

### 6.2.1 Sentence Classifier Training

Text classification is one of the common NLP tasks developed in the supervised machine learning field of study. A variety of classical algorithms and models are widely used to play the task (BAHARUDIN; LEE; KHAN, 2010), among them are Naive Bayes, Logistic

Regression, Supporting Vector Machines and Random Forests. Each one of them offers its advantages and disadvantages in use or application to solve specific problems as spam detection, categorization of news articles or sentiment analysis.

The sentence classifiers were applied in the extraction pipeline with two main objectives: first, to detect among sentences labeled as "other" on the initial datasets, those ones that may contain property values, in order to relabel them to "positive"; second and more importantly, to select candidate sentences for the attribute extraction, among all sentences in the candidate articles.

The sentence classifiers used in this work were built using Support Vector Machine (SVM) (CORTES; VAPNIK, 1995). It was trained one classifier for each property in the recovered schema. Baharudin, Lee and Khan (2010) have evaluated the use of well-known machine learning models for text classification and has stated that *"SVM has been recognized as one of the most effective text classification method in the comparison of supervised machine learning algorithms. It captures inherent characteristics of the data, independent of the feature space dimensionality"*. Although it presents some drawback in memory efficiency, SVM model has been effective in high dimensional spaces as text classification. Hence, pursuing better classification results we have opted for using SVM classification model, which also can be extended as in Chang and Lin (2011), to give probability estimates to update samples previously labeled as "other" by the *Distant Supervision* component.

After training the model for the given properties, the trained sentence classifiers is used to predict the sentences labeled as "other" by the DS method. This method is used to catch possible false positive sentences that were not matched by the hybrid DS assumption. Therefore, the sentences are passed through the trained classifier. If the prediction probability of being true is bigger than 90%, the sentences have their labels updated. The resulting datasets are used as training datasets for the information extractors.

### 6.2.2 Attribute Extractor Training

A popular technique applied for Information Extraction in natural language texts is the Named Entity Recognition tagging. The method is also known as entity identification, entity extraction or entity chunking. It consists of the identification of named entities in sentences and further classification under predefined classes.

Our work is based on NER, although it does not make use of generic categories as (*Person*, *Location*, *Organization*, and *Misc*) to label entities as traditional approaches in this area do. Instead, the tokens representing properties and values (to be extracted) are identified as "PROP", "VALUE", and "O" (from "other"). The sequence-based classification model is used to predict these categories on input examples.

Given the strong results provided by RNNs for sequence tagging, parsing, and other NLP tasks, as well as the application of CNNs for classification tasks (YOUNG et al., 2018).

**Source: Adapted from (CHIU; NICHOLS, 2016)**

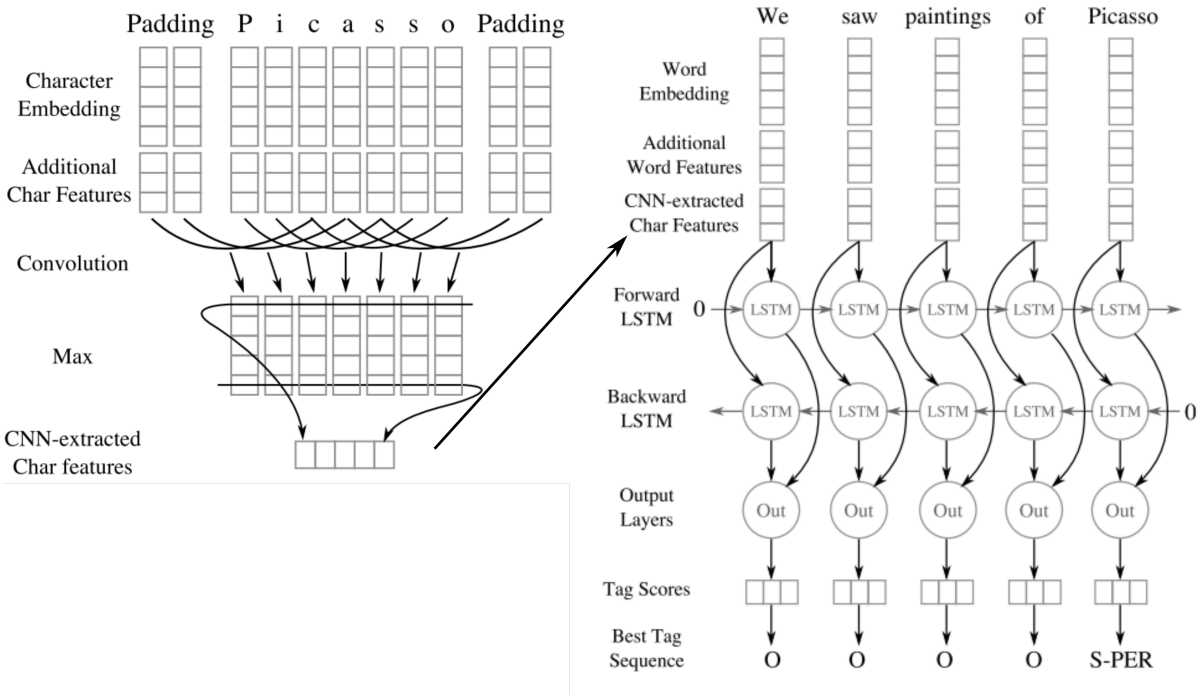



Figure 18 – LEFT.- The CNN extracts character features from each word. It creates embedding for each character. Padding tokens are added to the end and start of the word. The embeddings are concatenated, passed through the CNN and a max pooling layer. The output is the feature vector for the word at the character level. RIGHT - The BLSTM for tagging named entities. Pre-trained word embeddings, additional features word-level, and Character-level features obtained from the Convolutional Neural Network are concatenated for each word in the sentence. They are fed to the BLSTM network, and passed to the output layer. The output is the sequence tagging.

We have used a hybrid deep neural network model as the sequence-based classification model for this work. This model reduces the human involvement in feature engineering tasks as a CRF model would do.

While the character-level CNN allows the automatic extraction of features for characters composing a word, RNN model allows the sequence classification of tokens in a sentence (ELMAN, 1990). The variation of the RNN model used in this work is the BLSTM, which is composed by a combination of biRNN and LSTM. This combination allows the storage of context information for long contexts, and at the same time performs predictions of the current state looking forward and backward in the entered sequence. This behavior is important for our particular problem, since Wikipedia can present a variety of writing styles, containing short and long sentences.

The applied model receives as input a combination of pre-trained word embeddings, word features and character features obtained from a character-level CNN. The features are passed through a Bidirectional Long Short-Term Memory layer (BLSTM). The applied

deep neural network gives as output a sequence of predicted labels for sentence tokens received as input. The main objective of this hybrid architecture is to learn sentences, words, and character level features, and output the respective attribute extractor. The model architecture (see Figure 18) is based on Chiu and Nichols (2016), which have applied this architecture to the NER tagging task.

The character-level CNN (Figure 18) is used to extract character features from each word in the sentence. It makes use of a lookup table composed by numbers (0-9), special characters, and upper and lower letters (a-z, A-Z) to output a character embedding. The characters set also include tokens for *PADDING* and *UNKNOWN*, which are used for pad both sides of words according to the CNN window size, and to identify not listed characters in the lookup table, respectively. The additional (optional) character features used by the original model are *upper case*, *lower case*, *punctuation*, and *other*, used to represent the character type. Once, our vocabulary already includes these types of symbols, we decided not to use this additional character features. The character level features generated by the lookup table for each character in the word are concatenated, passed into the CNN, and finally by a max pooling layer to decrease the spatial size and output the word representation according to the character-level classification.

In addition character features extracted by the CNN, the hybrid neural network architecture (Figure 18) makes use of pre-trained word embeddings. The use of pre-trained word embeddings leverages the execution of the model, instead of training our own embeddings what would take too much effort given the size of the working corpus. We decided to use Stanford's GloVe which has been trained on 6 billion words from Wikipedia and Web text (PENNINGTON; SOCHER; MANNING, 2014). Glove 6B 50d have also shown competitive results for the CoNLL-2003 and OntoNotes, according to experiments done by Chiu and Nichols (2016). The model also makes use of additional word features: *allCaps*, *upperInitial*, *lowercase*, *mixedCaps*, *noinfo*. We have incremented it with *numeric*, *containsDigit* and *mainlyNumeric*. The model proposed by Chiu and Nichols (2016) makes use of lexicons as a form of external knowledge to help identify known named entities for the categories *Person*, *Location*, *Organization*, and *Misc*. We have decided to not use lexicons in the applied model, because our goal its not the NER tagging of such categories, but the sequence labeling of properties and values ("PROP", "VALUE", and "O"), which are general-purpose labels.

# 7 EXPERIMENTS

We have performed an extensive performance evaluation of our solution for unstructured information extraction on Wikipedia. Besides analyzing its overall performance, our goals included: examine baseline works for the same IE task, the dataset construction of each strategy and the proportion of filled properties from the defined schema. This chapter is organized into the experimental setup in Section 7.1, describing the datasets definition, applied schemas, evaluated approaches, and metrics; and the evaluation and results presentation in Section 7.2

## 7.1 EXPERIMENTAL SETUP

### 7.1.1 Datasets

To evaluate the performance of our work, we used 4 infobox templates: U.S. County, Artist, Airline and University (see Table 6). We have chosen them because they are among the 125 most-used templates on Wikipedia and 3 of them (U.S. County, Airline and University) were also used by previous approaches (WU; WELD, 2007; LANGE; BÖHM; NAUMANN, 2010). The attributes were selected using the strategy defined in chapter 6. Overall, a total of 29 attributes were chosen (Table 6).

The selected properties are expressed by different data types (numerical, textual, alphanumeric, and multivariate). The properties composition is also shown in Table 6. These different data types tend to make difficult the information extraction task. Numerical information is usually easier to detect and extract features than information mostly composed by textual and alphanumeric characters. The extraction of textual information is more sensitive to case, spellings, and writing styles variation (e.g., long and short sentences, writing structure descriptive or expository). Properties with alphanumeric symbols as letters and numbers are even more difficult to get a reliable pattern. For instance, properties related to datatypes (e.g., *founded*, *birth_date*, and *established*) can sometimes be represented by only a numerical chain (e.g., 2009, 1589), or as a data pattern composed by day, month, and year organized in different ways (e.g., *25 September 2009*; *April 29, 1882*; *c. 1589*).

The variation in the size of the values to be extracted as well as the size of the context in which they are usually found, are shown in 6. We have used the validation set to measure the average count of characters and sentences for each property. It is possible to identify that while some properties are present in minor contexts, e.g., sentences presenting the *density_sq_mi* have in average the count of tokens around 11.17, there are cases in which the information to be extracted is among a large context, e.g., sentences presenting information about *area* contain in average 33.36 tokens. Also, we can notice the differences

Table 6 – From left to right are presented: Infobox templates used as input to the *Schema Discovery* component; The schema size composed by most used properties in the domain; The data type expressed by each property (T - Textual, N - Numeric, A - Alphanumeric, M - Multivariate); The average count of tokens and characters for the values of each property; The average count of tokens in the sentences containing values for the respective properties; The count of tokens and characteres were obtained from the validation set.

| INFOBOX TEMPLATE | SCHEMA SIZE | DATA TYPE | DATASET | PROP. VALUE AVG. COUNTS | | SENT. AVG. COUNTS |
|---|---|---|---|---|---|---|
| | | | | TOKENS | CHAR | TOKENS |
| **U.S. County** | 12 | T | county | 2.12 | 14.28 | 14.88 |
| | | | largest_city_wl | 1.13 | 7.25 | 13.38 |
| | | | named_for | 3.75 | 22.25 | 25.36 |
| | | | seat_wl | 1.16 | 8.28 | 9.32 |
| | | | state | 1.16 | 7.78 | 14.26 |
| | | N | pop | 1.00 | 5.05 | 19.52 |
| | | | density_sq_mi | 1.00 | 2.63 | 11.17 |
| | | | area_land_sq_mi | 1.00 | 3.44 | 33.36 |
| | | | area_total_sq_mi | 1.00 | 3.48 | 33.36 |
| | | | area_water_sq_mi | 1.00 | 2.58 | 33.36 |
| | | A | area_percentage | 2.00 | 4.10 | 33.36 |
| | | | district | 1.00 | 3.00 | 24.00 |
| **Airline** | 7 | T | airline | 2.38 | 14.54 | 26.00 |
| | | | callsign | 1.00 | 7.50 | 20.00 |
| | | | headquarters | 6.75 | 39.83 | 26.00 |
| | | | IATA | 1.00 | 2.50 | 20.50 |
| | | | ICAO | 1.00 | 2.00 | 4.00 |
| | | M | fleet_size | 1.00 | 2.00 | 10.40 |
| | | A | founded | 1.08 | 4.83 | 22.67 |
| **Artist** | 5 | T | birth_place | 2.64 | 12.82 | 21.45 |
| | | | name | 3.00 | 17.88 | 22.75 |
| | | | nationality | 1.08 | 7.46 | 22.77 |
| | | M | field | 3.00 | 19.79 | 23.00 |
| | | A | birth_date | 2.13 | 8.94 | 22.88 |
| **University** | 5 | T | city | 1.86 | 11.14 | 26.57 |
| | | | country | 1.14 | 6.14 | 25.86 |
| | | | name | 7.86 | 50.86 | 25.57 |
| | | M | type | 1.00 | 6.43 | 25.43 |
| | | A | established | 1.29 | 6.29 | 18.71 |

**Source:** This research

in the size of the information to be extracted, while the majority of the properties present an average information to be extracted below 2 tokens, there are cases in which the information to be extracted is almost a sentence, e.g., University's *name* (7.86 tokens), Airline's *headquarters* (6.75 tokens), and U.S County's *named_for* (3.75 tokens).

When dealing with this number of properties to be extracted, there are cases in which

the property presents a high frequency on the infobox instances, but its value is rarely found in the articles text, e.g., *IATA* and *ICAO* (airport codes). Also, there is the case of *fleet_size*, which is present in a considerable number of infobox instances but is not present in the articles text as a unique concrete value. This is mostly because the information present in the text is organized by categories of aircraft models, separated by groups and counted, i.e., instead of the information be expressed as in the sentence *"...the fleet size is of 5 aircrafts..."*, it is common to be presented as *"...mainline fleet consists of the following aircrafts: 2 Boeing 737, 3 Boeing 767-300ER..."*. Therefore, the property most times represents an information to be inferred.

The different sizes of the schema (Table 6) may infer the popularity of the class or the lack of structured information existent in the Wikipedia. It can also indicate specialized domain. For instance, while the selected schema for *Artist* and *University* is mostly composed by generic properties (e.g., *city*, *country*, *type*, *nationality*, *birth_date*, *birth_place*), the schema for U.S County and Airline presents specific properties as *IATA*, *ICAO*, *callsign*, and information related to area, seat, density, and population.

Based on previous approaches for the task, e.g. Kylin and iPopulator, 100 articles were randomly selected for each infobox template. A set of 50 articles were labelled and used as validation, and another set of 50 articles were labeled for test. Each article was eyeballed, and property values were manually extracted to construct the validation and test sets for each schema property.

### 7.1.2 Approaches

To compare our work with previous ones, we have implemented each baseline approach from the ground. We executed the following approaches for comparison:

Table 7 – Features set used by CRF extractor.

| Feature Description | Example | Feature Description | Example |
|---|---|---|---|
| First token of sentence | *Hello* World | Contains an underline symbol | km_square |
| In first half of sentence | *Hello* World | Contains an percentage symbol | 20% |
| In second half of sentence | Hello *World* | Stop word | the; a; of |
| Start with capital | Hawaii | Purely numeric | 1929 |
| Single Capital | A | Number type | 1932; 1,234; 5.6 |
| Start with capital, end with period | Mr. | String normalization: capital to "A", lowercase to "a", digit to "1", others to "0" | TF—1 $\Longrightarrow$ *AA*01 |
| All capital, end with period | CORP. | Part of Speech tag | |
| Contains at least one digit | AB3 | Token itself | |
| Made up of two digits | 99 | NP chunking tag | |
| Made up of four digits | 1999 | Previous tokens (window size 5) | |
| Contains a dollar sign | 20$ | Following tokens (window size 5) | |

OBS: Features for anchored text and tokens present on original work were discarded.

**Source:** Adapted from (WU; WELD, 2007)

- **Kylin**: as described in Chapter 3, it is a system proposed by Wu and Weld (2007) that looks for autonomously creation and completeness of Infoboxes. It builds sentence classifiers using the sentence tokens and Part of Speech (POS) tags as features to train a Maximum Entropy Model. It employs a bagging ensemble strategy to overcome noisy, incomplete and unbalanced data. To build extractors, it makes use of a CRF model with the features described in Table 7. As mentioned in the Table 7 footnote, while training the CRF models, we have discarded original features related to anchored text and tokens, once our recovered and processed data do not contain any anchor or hyperlink information. We have found some difficulties while trying to implement the system. First, the work has not mentioned how articles and infoboxes were processed, to handle the text information. As mentioned in Chapter 4, a Wikipedia page content can be gathered through Wikicode although there is always the option to crawl the HTML pages. The second problem was to apply the bagging of Maximum Entropy models as implemented by Mallet[1] for classification, mainly because of the framework poor documentation. Looking to precisely follow the original work without any obstacles, we have changed the classifier model implementation for a more recent one, the Bagging Classifier ensemble method and

[1] <http://mallet.cs.umass.edu/topics.php>

Table 8 – Features used by iPopulator to train CRF extractors.

| Description |
|---|
| **Length** |
| Small token (length < 10) |
| Within long paragraph (#sentences > 2) |
| **Relative token positions** |
| Position in article: in 1st, 2nd, 3rd third |
| Position in paragraph: in 1st, 2nd, 3rd third |
| Position in sentence: in 1st, 2nd, 3rd third |

(a) Features for the currently analyzed token only

| Description | Example |
|---|---|
| **General** | |
| Value of the token | 2010 |
| Type | Number, String |
| Part-of-speech tag | NN |
| Enclosed by formatting symbols | [”] Hi [”] |
| Enclosed by the formatting symbols ”’? | [”’] Title [”’] |
| **Structure** | |
| Two-digit number | 17 |
| Four-digit number | 2010 |
| Number | 1511 |
| Formatted numbers (thousand sign) | 1,121 |
| Formatted numbers (thous., mill./bill.) | 1 mill. |
| Alphanum. char. (start: letter) | Company |
| Alphanum. char. (start: letter, end: dot) | end. |
| **Check for Occurrence** | |
| Token km/mi/miles | km |
| Token contains “http” | http://faz.de |
| Token contains “?” | Who? |
| Token contains “.” | him. |

(b) Features for all tokens inside a window of five tokens before and after the currently analyzed token.

**Source:** (LANGE; BÖHM; NAUMANN, 2010)

Logistic Regression model from Scikit Learning library[2]. Kylin has also used the Mallet implementation to build the CRF model. In our implementation of Kylin, we have also changed the model implementation for the SKlearn CRFSuite[3] with Limited-memory Broyden–Fletcher–Goldfarb–Shanno (L-BFGS)(BYRD et al., 1995) feature weight estimation method.

- **iPopulator** As shown in Chapter 3, iPopulator is a rule-based system proposed by Lange, Böhm and Naumann (2010). It first applies rules to get patterns that represent the syntactic structure of the property values, this step is called *Structure Analysis*. Dealing with a rule-oriented system like iPopulator, brings some difficulties. In this case, the main difficulty is related to the parsing function to detect value attributes structure. The work does not clearly list the types of tokens to be detected as syntactic structures. Hence, we have used the following three type detectors: text, number, and alphanumeric. These detectors were used to infer patterns for each property value occurrence. The obtained patterns are sorted, ranked and then merged to compose a final result expression. During the *Training Data Creation* it applies a Fuzzy matching strategy to sequence label the sentence tokens. The fuzzy matching is composed of two functions, one for textual values and the other for numeric values. The similarity function for textual parts

[2] <scikit-learn.org>
[3] <https://sklearn-crfsuite.readthedocs.io/en/latest/api.html>

apply the Levenshtein edit distance, using a similarity threshold empirically defined ($0.001 * length(first\_token)$). The similarity function for numeric values calculates the absolute difference between compared tokens and compares the result to the defined threshold ($0.001 * fist\_token$). The labels used for sequence tagging the sentences are the position values of the respective structure part previously defined, e.g., pattern **(Number '(' Number ')')** for value "12,500 (2003)" provide labels 1 and 2 to respectively matched sentence tokens). iPopulator also applies a CRF model to lead the extraction process. We have noticed that the features applied to train the CRF model were rather simplistic (See Table 8). While training value extractors all of them are automatically evaluated, and only the ones presenting precision above 0.75 are kept. This elimination is performed to boost precision over recall. During the execution of experiments shown in the next sections, we have maintained extractors for all evaluated attributes, to conduct fair comparisons between models. Lastly, when looking for properties extraction iPopulator make use of only a few first paragraphs of the article text. We have used the first five paragraphs.

- **Deepex-BLSTM+CNN** It corresponds to our system described in Chapter 6. Due to automatic dataset construction done with the assistance of distant supervision and similarity measures, the generated initial datasets to train sentence classifiers have shown unbalanced numbers of negative and positive samples (See Table 13). Looking for computational efficiency and better fit of the data, we have applied undersampling over the training set to train the sentence classifiers. It was applied a random undersampling strategy to remove samples from the majority class. The training of sentence classifiers consists in the use of BoW vectors features with TF-IDF weighting. We have used the Sklearn SVC[4] implementation of SVM with the default Radial Basis Function (RBF) kernel, and enabling the probability estimate. Because of the vast number of attributes to be evaluated, and the manual verification employed over the constructed validation set, it would require extensive human effort fit different parameters to all datasets, becoming a time-consuming task. Hence, instead of define different hyper-parameters for each property dataset, we have used the validation set of 50 articles to empirically define some hyper-parameters (Table 9) common to all the generated datasets. For that, we have used the *U.S. County* domain, because of its less amount of training data and its variety of properties encompassing different data types. The defined Deep Neural Network architecture is composed by one layer of BLSTM with recurrent dropout of 0.25. The entire model was trained with the *NAdam* optimizer. The number of epochs (10) for training was defined using an early stopping strategy after 5 iterations with no significant improvement over the validation set. We have trained the extractor model using only tagged sentences passed through sentence classifier and predicted as containing

[4] <https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html>

properties value. These sentences are coming from the initial DS matching described in Chapter 6.1.

Table 9 – Hyper-parameter values applied for training neural network architecture used in this work.

| **Hyper-parameters** | **Value** | **Search space** |
|---|---|---|
| Convolution Width | 3 | [3, 5] |
| CNN output size | 50 | [10, 50] |
| LSTM state size | 275 | [100, 300] |
| LSTM dropout | 0.68 | [0.25, 0.75] |

**Source:** This research

- **Deepex-BLSTM_W2** It consists of a variation of the proposed architecture for extractors in Chapter 6.2.2. It makes use of the BLSTM layer aligned only with pre-trained word embeddings, and word features. It makes use of the same hyper-parameters defined for the Deepex-BLSTM+CNN (see Table 9);
- **Deepex-BLSTM** It consists of another variation for the proposed architecture for extractors in Chapter 6.2.2. It makes use of the BLSTM layer aligned only with pre-trained word embeddings. It makes use of the same hyperparameters defined for the Deepex-BLSTM+CNN (see Table 9);
- **Deepex-CRF** Our work has also investigated the use of CRF in our extraction pipeline defined in Chapter 6. Hence, we have replaced the hybrid neural network of BLSTM and character-level CNN layers by a CRF model. The applied CRF model has used the same features described on Table 7 for training Kylin extractors. The implementation of Sklearn CRFSuite[5] with L-BFGS (BYRD et al., 1995) feature weight estimation method were also used to model this CRF extractor.

The four Deepex variations of the extraction model were independently embedded in our pipeline. The experiments were executed for each combination of the pipeline and extractor models. The benchmark between the four models and baseline works can be seen in Section 7.2.

The difficulty in comparing iPopulator and Kylin is highlighted by Lange, Böhm and Naumann (2010). iPopulator can be evaluated in a fine-grained measure, due to the partitioning of values to be extracted, on the other hand, Kylin extracts complete values. In Lange, Böhm and Naumann (2010) the evaluation settings used to compare the two models were different, what can lead to an imprecise performance comparison (e.g., evaluated properties, the selection, and number of Articles used to compose validation set).

[5] <https://sklearn-crfsuite.readthedocs.io/en/latest/api.html>

To overcome the comparison problems highlighted by Lange, Böhm and Naumann (2010), when comparing these different models we have established accurate criteria to compute metrics. Different from Lange, Böhm and Naumann (2010), we have run the experiments applying the same scenario to all evaluated models, i.e., using the same datasets, infobox templates, evaluating its performance over the complete schema, without attribute elimination as it is the case of Lange, Böhm and Naumann (2010).

### 7.1.3 Metrics

We have analyzed the systems performance through the computation of precision, recall, and f-score of each evaluated schema property. Given the different approaches for extraction, mainly from iPopulator fragmented extraction, and multivalued properties, we have performed a partial matching between expected and extracted outputs. We have computed the Jaccard similarity to measure the distance between the strings and differ positive from negative matchings, e.g. the value *"MacAir Airlines Pty"* is a True Positive (TP) extraction for the expected value *"MacAir Airlines Pty Ltd"*. However, exact matching would indicate it as a False Positive (FP). To compute the metrics we have applied the following definitions:

- True Positive (TP): extracted values have matched expected output. For atomic values, its when the Jaccard distance is below or equal to 0.5, e.g., expected value *"Major General Patrick R. Cleburne"* presents a Jaccard distance of 0.2 from the extracted value *"Patrick R. Cleburne"*, thus, it is a TP. For the cases of multivalued and iPopulator extractions, is computed the Jaccard similarity between each extracted value and the expected one(s). If the distance is below or equal to 0.5 it indicates a positive matching. It is taken the proportion of positive matchings from the total of values extracted, if the proportion of positive matchings is above 0.5, we have a true positive extraction. For instance, the expected multivalue *"photography, painting, sculpture'* and the extracted value *"photography, painting"* presents a proportion of 0.66 of positive matching.

- False Positive (FP): a value is extracted when it is not expected extraction, and when the extracted values are different from the expected output. For atomic values is when the Jaccard distance is above 0.5. For multivalued and iPopulator extractions is when the proportion of positive matchings is below 0.5.;

- True Negative (TN): the model does not perform extraction when no extraction is required;

- False Negative (FN): the model does not perform extraction when extraction is required.

We have computed Precision, Recall and F-score for each dataset. However, to better evaluate extractions according to extracted infobox templates, we have focused the evaluation in the micro and macro Precision, Recall and F-score measures. The macro average measures the overall performance of the systems across different datasets for one given infobox template. The micro measure aggregates the contributions of all datasets from the infobox template. Therefore, for simplicity, when analyzing the performance of the models over a given Infobox template, we will be referring to macro and micro measures, otherwise, when referring to the performance of the models over separate property datasets we will be referring to common Precision, Recall and F-score measures. The Table 14 for complete macro and micro Precision, Recall, and F-score measures by Infobox Template can be found in the Appendix A, and the Table 17 to all computed F-score measures ffor each dataset in Appendix D. To clear differences in performance across evaluated infobox templates, we have computed the micro and macro measures for each one.

Since the baselines have not clarified how articles text and infobox information were preprocessed, also considering constant evolving of Wikipedia structures, articles, and event writing styles, it is expected that obtained baseline results will differ from the original works. We have focused on measuring the performance of the systems as a whole, using extracted and expected outputs as parameters to measure precision, recall, and F-score.

## 7.2 EVALUATION AND RESULTS

The macro and micro F-score values obtained by all approaches in the 4 domains are presented in Table 10. All the variations of Deepex obtained better results than the two baselines (Kylin and iPopulator) both in terms of macro and micro F-score. To verify if these results are statistically significant, we have used the F-scores obtained for each dataset (property), see Table 17 in Appendix D, and executed the Shapiro normality test (SHAPIRO; WILK, 1965) in the distributions of each approach. The normality test have rejected the null hypothesis for normality in all distributions. Hence, we have executed the Wilcoxon Signed-rank Test (WILCOXON, 1945) on the difference of the F-scores between all approaches (See Table 11) to verify the following hypothesis:

$H_0$ There is no difference between the models.

$H_1$ There is significant difference between the models.

Table 10 – Macro and Micro F-score by model for each class

| F-SCORE | | U.S. county | Airline | Artist | University |
|---|---|---|---|---|---|
| **Macro** | Kylin | 0.5053 | 0.2791 | 0.3256 | 0.4233 |
| | iPopulator | 0.5328 | 0.2972 | 0.3247 | 0.3907 |
| | DEEPEX-CRF | 0.8109 | 0.3770 | **0.8009** | 0.7349 |
| | DEEPEX-BLSTM+CNN | **0.8230** | 0.3431 | 0.7656 | 0.7818 |
| | DEEPEX-BLSTM_W2 | 0.7923 | **0.4246** | 0.7471 | **0.7867** |
| | DEEPEX-BLSTM | 0.8004 | 0.3840 | 0.7666 | 0.7811 |
| **Micro** | Kylin | 0.6535 | 0.4651 | 0.4548 | 0.5049 |
| | iPopulator | 0.5990 | 0.5000 | 0.4035 | 0.4191 |
| | DEEPEX-CRF | 0.8344 | 0.5866 | **0.8000** | 0.7374 |
| | DEEPEX-BLSTM+CNN | **0.8598** | 0.5290 | 0.7638 | 0.7891 |
| | DEEPEX-BLSTM_W2 | 0.8274 | **0.5905** | 0.7462 | **0.8010** |
| | DEEPEX-BLSTM | 0.8270 | 0.5625 | 0.7631 | 0.7951 |

**Source:** This research

In the Table 11, $N$ is the sample size (it were used 29 datasets), for the test execution the ties in the distribution are discarded,once $N$ is bigger than 20 in all cases, we can use the *p-value* to evaluate our hypothesis. The numbers show that all Deepex variations obtained statistically superior results in comparison with the baselines. This can be explained by the restrictive nature of their heuristics to build the training datasets for extraction, as Table 13 and 12 depict.

In the better cases Kylin have achieved macro F-score for U.S. County and University of 0.5053 and 0.4233, respectively. iPopulator have achieved 0.5328 and 0.3907 for the same domains. In some cases Kylin's performance was impaired by the datasets construction, as shown in Table 13 some properties for the Kylin pipeline could not leverage a sufficient amount of training samples, e.g., *county*, *district*, *pop*, and *headquarters*. For the property *district* in the domain U.S. County, our approach extracted 1.584 positive examples whereas Kylin only 616 and iPopulator 528. Regarding negative examples, the difference is even bigger (our approach: 144.906, Kylin: 0). This problem have implicated in the coverage of extracted properties.

As shown in Table 12, the proportion of extracted properties is affected by Kylin's heuristics in the 4 domains. Kylin was not able to fill any schema completely, different from the Deepex variations and even iPopulator. This confirms the observation of restrictive heuristics. For the U.S. County domain, for instance, Kylin was able to extract only 67% of the properties whereas iPopulator 92%, and Deepex 100%. The highest proportion coverage of Kylin comes from the extractions in the U.S. County domain, which presents a large amount of numerical properties. Therefore, Kylin is restricted to correctly extract

Table 11 – Results for the Wilcoxon Signed-rank Test.

| **Comparison** | **N** | **p-value** | **Null Hypothesis** ($\alpha = 0.01$) |
|---|---|---|---|
| **Kylin vs DEEPEX** | | | |
| Kylin vs Deepex-CRF | 27 | $3.20 \cdot 10^{-4}$ | Rejected |
| Kylin vs Deepex-BLSTM | 27 | $1.60 \cdot 10^{-4}$ | Rejected |
| Kylin vs Deepex-BLSTM_W2 | 26 | $2.40 \cdot 10^{-4}$ | Rejected |
| Kylin vs Deepex-BLSTM+CNN | 27 | $1.80 \cdot 10^{-4}$ | Rejected |
| **iPopulator vs DEEPEX** | | | |
| iPopulator vs Deepex-CRF | 27 | $3.20 \cdot 10^{-4}$ | Rejected |
| iPopulator vs Deepex-BLSTM | 27 | $1.40 \cdot 10^{-4}$ | Rejected |
| iPopulator vs Deepex-BLSTM_W2 | 28 | $1.60 \cdot 10^{-4}$ | Rejected |
| iPopulator vs Deepex-BLSTM+CNN | 28 | $1.40 \cdot 10^{-4}$ | Rejected |
| **DEEPEX CRF vs DL** | | | |
| CRF vs BLSTM | 24 | $9.92 \cdot 10^{-1}$ | Accepted |
| CRF vs BLSTM_W2 | 25 | $7.28 \cdot 10^{-1}$ | Accepted |
| CRF vs BLSTM+CNN | 25 | $7.79 \cdot 10^{-1}$ | Accepted |

**Source:** This research

information from numerical properties, and for this reason underperforms the extraction of textual properties. It achieved high F-score results only for textual properties with a well defined structure in the articles text, e.g., artist's *name* (0.85) and *birth_date* (0.77), university's *name* (0.94), and *named_for* (0.68). Another issue with Kylin's heuristics to build the training data that may have caused the low properties coverage: the low number of labeled false samples. Even though Kylin applies a Bagging strategy to decrease the impact of noisy and incomplete data, the lack of representation for negative sentences is evident.

Regarding iPopulator, it was not able to extract information only for the properties: *district*, *IATA*, and *ICAO*. All these properties represent information rarely seen on articles text. Despite the fact that iPopulator have managed to extract information for almost all properties in the schema, the macro and micro results presented in Table 10 indicate the lack of Precision and Recall in the extractions (See Table 14 in Appendix A for macro and micro Precision and Recall measures). This may have happened because iPopulator restricts its search for properties to only a few first paragraphs from the Articles text,

Table 12 – Proportion of filled properties in defined schema.

| | **U.S. county** | **airline** | **artist** | **university** |
|---|---|---|---|---|
| KYLIN | 0.67 | 0.57 | 0.40 | 0.60 |
| IPOPULATOR | 0.92 | 0.71 | 1.00 | 1.00 |
| DEEPEX-CRF | 1.00 | 0.71 | 1.00 | 1.00 |
| DEEPEX-BLSTM | 1.00 | 0.71 | 1.00 | 1.00 |
| DEEPEX-BLSTM_W2 | 1.00 | 0.71 | 1.00 | 1.00 |
| DEEPEX-BLSTM+CNN | 1.00 | 0.71 | 1.00 | 1.00 |

**Source:** This research

e.g., properties like *density* and *district* are rarely found in the first paragraphs of the text. The labeling of property values fragment with the use of fuzzy matching have also generated noisy training data.

The iPopulator system, unlike Kylin, presents a vast number of labeled sentences to train extractors. There are cases where the count of positive samples in iPopulator is about two times bigger than in Kylin datasets, or even more, e.g.: University's *city* (iPopulator: 33543, Kylin: 14594), *name* (iPopulator: 68158, Kylin: 33662). The average number of positive samples clearly shows the difference: iPopulator (9,769.90), Kylin (6,856.38). This behavior may have caused the labeling of a high number of wrong samples, which have decreased the performance of the extractors. A large amount of samples in iPopulator datasets indicates that its heuristics are widely open to get multiple fragments of the same value attribute through the use of fuzzy matching, see the example below:

**Tuple:** (birth_date, 1827 / 11 / 03)
**Sentence:** "Alexander Fraser was born at Woodcockdale, near Linlithgow on **3 November 1827** and was baptised on 11 January 1828 at Linlithgow."

**Labels:** ['O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', '**3**', 'O', '**1**', 'O', 'O', 'O', 'O', '**2**', 'O', '**1**', 'O', 'O', 'O']

In the above example iPopulator have matched only the day and the year of the birth, and could not match the month. Furthermore, because of the fuzzy matching thresholds, it has also matched the day and year of baptism as if they represent the birth date. This wrong labeling of iPopulator can also be extended for the cases of multivalued attributes and lists of values. This flexibility should increase the extraction recall, the amount of noisy data, and consequently decrease the precision and F-score.

Deepex comes as a mid-term between Kylin and iPopulator. Its Distant Supervision output presents a considerable amount of negative and positive training examples, in

average 7,202.24 positive examples and 175,773 negatives (See Table 13). As stated before, the unbalancing of the datasets are circumvented by the undersampling during sentence classifiers training. As shown in table 12, the Deepex variations is able to fill the entire schema from three infobox templates. The properties *fleet_size* and *IATA* from infobox template Airline, as aforementioned, are properties rarely seen on articles text, which makes difficult its extraction by the models.

When comparing the Deepex version with CRF versus Deepex variations with Deep Learning architectures, the *p-values* for the hypothesis test (Table 11) are always above the level of significance, accepting the null hypothesis. For these cases we can state that applied Deep Learning (DL) architectures have shown the same performance for the task as the widely used CRF model. However, the DL variations of Deepex have performed better for some cases as numerical and textual properties.

The Deepex variation using CRF presented better macro and micro results for the Artist infobox template, 0.8009 and 0.8000, respectively. The properties for Artist template present the mixed data type *birth_place*, the multivalued property *fields* (an artist can be assigned to various fields), and textual properties that do not present long chains of text (*name* and *nationality*). In spite the different data types from properties schema, the Deepex-CRF have outperformed the others for the textual and multivariate attributes, possibly because of the well-defined CRF features (Table 7), and of the simple and direct writing style of the articles under this domain. However, it have underperformed the other DL aproaches for the alpha numeric property.

The Deepex-BLSTM_W2 have outperformed all the other models in two datasets, Airline with the macro F-score of 0.4246, and University with macro F-score of 0.7867. The writing style in these domains is rather flexible and specialized, presenting a more challenging extraction. Both templates have properties related to textual and alphanumeric data types, representing date information (*founded*, and *established*). While the textual information to be extracted can present long chains of text (e.g., *headquarters* from Airline, *named_for* from U.S. County, and *name* from University). As shown in Table 6, these properties are composed by long chains of words and characters, *headquarters* with an average of 6.75 tokens, *named_for* with 3.75, and *name* from University with 7.86.

Table 13 – The output of the Distant Supervision. Labels count for each schema property according with evaluated model.

| CLASS | PROPERTY | KYLIN | | IPOPULATOR | DEEPEX | | |
|---|---|---|---|---|---|---|---|
| | | Pos. | Neg. | Pos. | Neg. | Pos. | Other |
| US County | area_land_sq_mi | 2332 | 90 | 2803 | 54071 | 4216 | 96275 |
| | area_percentage | 3214 | 65 | 3042 | 97466 | 1308 | 54947 |
| | area_total_sq_mi | 2253 | 86 | 2817 | 91245 | 4043 | 59329 |
| | area_water_sq_mi | 3189 | 2968 | 3169 | 98702 | 5160 | 50466 |
| | county | 28299 | 0 | 13820 | 71034 | 7423 | 76349 |
| | density_sq_mi | 1642 | 2446 | 1014 | 115277 | 4222 | 28265 |
| | district | 616 | 0 | 528 | 144906 | 1584 | 7748 |
| | largest_city_wl | 746 | 15547 | 4849 | 93537 | 1941 | 48667 |
| | named_for | 1998 | 664 | 1438 | 54895 | 2511 | 58238 |
| | pop | 23 | 4 | 739 | 124034 | 7073 | 23570 |
| | seat_wl | 6137 | 11625 | 5324 | 111406 | 3792 | 29619 |
| | state | 7717 | 15811 | 7337 | 113290 | 15495 | 25994 |
| Airline | airline | 17212 | 4310 | 11035 | 31489 | 4418 | 57510 |
| | callsign | 3954 | 2844 | 1985 | 76379 | 635 | 7433 |
| | fleet_size | 1546 | 4660 | 385 | 59075 | 1226 | 18711 |
| | founded | 2009 | 1314 | 4502 | 57255 | 3074 | 29002 |
| | headquarters | 264 | 518 | 3991 | 27112 | 1846 | 60510 |
| | IATA | 6085 | 2139 | 303 | 80517 | 391 | 3413 |
| | ICAO | 3183 | 3296 | 1024 | 79084 | 236 | 5360 |
| Artist | birth_date | 3331 | 1814 | 11 | 340963 | 5515 | 155645 |
| | birth_place | 872 | 4940 | 6658 | 178902 | 6868 | 305377 |
| | field | 1292 | 4016 | 24576 | 221745 | 9556 | 192083 |
| | name | 25016 | 1191 | 25158 | 294210 | 12493 | 209569 |
| | nationality | 3278 | 27804 | 15624 | 367670 | 8044 | 67230 |
| University | city | 14594 | 60612 | 33543 | 474681 | 13517 | 178201 |
| | country | 10959 | 38616 | 14475 | 391257 | 23267 | 220564 |
| | established | 10607 | 10260 | 14459 | 533380 | 23086 | 119102 |
| | name | 33662 | 676 | 68158 | 236380 | 29420 | 403964 |
| | type | 2805 | 3584 | 10560 | 477455 | 6505 | 151380 |
| **AVERAGE** | | 6856.38 | 7651.72 | 9769.90 | 175773.00 | 7202.24 | 94638.66 |

**Source:** This research

For the Airline template, Deepex-BLSTM_W2 have presented high F-score for the properties *founded* (0.80), *headquarters* (0.62), and *callsign* (0.25). The *callsign* property is a textual information rarely found on articles text, and is mostly presented as *all caps*, a word feature included in the extractor architecture. The BLSTM have captured context in long chains of text, allowing the extraction of long property values as *headquarters*, and University *name* which have obtained a F-score of 0.96. Hence, given the exposed we can ensure that the word features added to the BLSTM network using word embeddings have made easier the extraction of information from these especialized domains, and with flexible writing styles. Adding the CNN to the BLSTM model have increased the extraction recall, and decreased the performance of the extractor. This can be caused by the learning of too many features from of unusual sentences containing the target

information.

Lastly, the Deepex-BLSTM+CNN have presented high extraction results for the U.S. County template, macro F-score of 0.8230 and micro of 0.8598. The properties belonging to this domain are composed by numeric, textual and alphanumeric data types. Different from Artist and University, the articles from the U.S. County domain present a well-structured writing style, with standardized sentences presenting important information. This information domain presents a lot of numerical properties (*area_land_sq_mi*, *area_total_s_mi*, *area_water_sq_mi*, *density_sq_mi*, *pop*), that can have its features easily extracted by a character level CNN, once it will treat only with numbers an puntuation. The template also presents properties related to places as *country*, *state*, *seat_wl*, *largest_city_wl*, *district*. The pre-trained word embeddings used in this network may contain semantic information of these target words that have increased the preformance of the model. Also, the use of this dataset and model to fit hyper-parameters may have affected the superior performance of Deepex-BLSTM+CNN under the U.S. county domain.

# 8 CONCLUSIONS AND FUTURE WORK

This chapter finishes our work listing all contributions, limitations, future works that can still be done to improve it, and concludes answering the raised research questions. Section 8.1 presents a brief discussion about the differences and similarities between our work and the baselines. In Section 8.2 are discussed the contributions and limitations of this work. In Section 8.3 are shown perspectives and opportunities for future works in the field. The last Section finishes the work by answering research questions.

## 8.1 DISCUSSION

Kylin (WU; WELD, 2007), and iPopulator (LANGE; BÖHM; NAUMANN, 2010) are also methods for Wikipedia attributes extraction. These works present similar approaches to extract unstructured information to populate/create infoboxes autonomously. However, their models rely on specifics heuristics and feature engineering to build training datasets and train models.

While Kylin presents strict heuristics to build training datasets, and sentence classifiers to filter the passage of sentences to the extractor, iPopulator makes use of broad huristics to build training datasets, and fuzzy matching to differ between numerical and textual information. The work presented here has merged both paths, using distant supervision along with similarity measures between attributes names, values and sentence tokens to build training datasets. We have also used sentence classifiers to filter the flow of sentences passed to the attribute extractor.

Different from Kylin and iPopulator, which makes use of the traditional sequence-based model CRF for NER tagging and further entity extraction, we have applied the hybrid architecture of BLSTM aligned with character level CNN, designed by Chiu and Nichols (2016), to automatically extract character and word features from sentences, and work as a NER tagger to guide the information extraction process.

## 8.2 CONTRIBUTIONS AND LIMITATIONS

Starting by the contributions, we can list the followings:

1. First, an indexing architecture is presented in order to explore Wikipedia corpus and DBpedia RDF turtle files, and also to assist the analysis of Wikipedia structure, data distribution, heterogeneity and quality. The data analysis is an important step to support schema and domain discovery through Categories hierarchy;

2. Second, the modifications of the Distant Supervision assumption have allowed an effective labeling of properties and values from different spellings and formats between supervision source and plain texts;

3. Third, we have showed that the use of Deep Learning architectures for NER tagging can present competitive performance to the state-of-the-art CRF model; and

4. Fourth, our proposed extraction pipelines have surpassed baseline works for the same problem, the slot-filling task on Wikipedia corpus.

An important limitation of our work is the absence of differential treatment to label multivalued properties. To easily evaluate all properties in suggested schema we have treated multivalued properties with an "OR" approach.

## 8.3 FUTURE WORK

Although we have improved the extraction of property values on Wikipedia corpus, some points and work can still be done to improve information extraction on an unstructured corpus. As a first step, we plan to investigate strategies to remove noisy labeling from the Distant Supervision assumption. The work presented here have applied an approach based on similarity measure to detect mentions of properties and values in the sentences, but it can still apply wrong labels.

For sentence labeling we intend to study the use of topic-based models and pattern correlations, as in the works of Alfonseca et al. (2011), and Takamatsu, Sato and Nakagawa (2012). An important improvement to our extraction pipeline is the extraction of multivalued properties. Similar to Lange, Böhm and Naumann (2010), we intend to investigate the use of hidden variables to learn property values pattern (atomic or multivalued), to include it to assist the extraction process. Also, we intend to use neural networks to model sentence classifiers, that are currently built with SVM classifiers.

The automatic definition of a target schema based on Categories is a good strategy to explore hierarchical and tagged structures in the Web. In this work, we could highlight some important aspects of this data organization, although investigations still need to be done to provide an autonomous schema definition. Beyond data analysis, it is possible to apply supervised and unsupervised learning methods to the task.

Finally, as a form to validate our strategy when scaling to the diversity of Web, we intend to apply it to other corpus, i.e., datasets minerated from Web sources like sites of real estate for sale, web forums, products specifications from e-commerce sites, among others.

## 8.4 CONCLUSION

Given the exposed, and the initially raised research questions, we can conclude:

1. Although we could not define a common schema for Categories, the data analysis performed on Wikipedia structures have shown that the study of the Categories hierarchy can assist the information domain and schema discovery, allowing a better understanding of information distribution across categories. Categories are widely used on the Internet, sometimes called as tags. It can be found not only in Wikipedia, but also in Web forums, blogs and news platforms;

2. The Soft TF-IDF similarity can be applied to modify the Distant Supervision assumption. It allows the matching of tokens from the supervision source and plain texts, considering word miss-spellings or formats without generate a high amount of noise;

3. Extraction experiments done with our proposed extraction pipeline (modifying the sequence tagging model to use CRF and variations of a Deep Learning architecture) have shown that even though CRF model got high F-score for two infobox classes (airline and artist), the differences between the four Deepex variations are not statistically significant. Therefore, a Deep neural network architecure can be competitive with the widely used CRF model for the sequence classification problem of NER tagging.

# REFERENCES


AGGARWAL, C. C.; ZHAI, C. X. *Mining text data*. Boston, MA: Springer US, 2013. 1–522 p. ISBN 9781461432234.

AGICHTEIN, E.; GRAVANO, L. Snowball: Extracting Relations from Large Plain-Text Collections. *Proceedings of the 5th ACM International Conference on Digital Libraries*, I, n. 58, p. 85–94, 2000. ISSN 158113231X.

ALFONSECA, E.; FILIPPOVA, K.; DELORT, J.-Y.; GARRIDO, G. Pattern Learning for Relation Extraction with a Hierarchical Topic Model. *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2*, n. July, p. 54–59, 2011.

APROSIO, A. P.; GIULIANO, C.; LAVELLI, A. Automatic expansion of dbpedia exploiting wikipedia cross-language information. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, v. 7882 LNCS, p. 397–411, 2013. ISSN 03029743.

APROSIO, A. P.; GIULIANO, C.; LAVELLI, A. Automatic Mapping of Wikipedia Templates for Fast Deployment of Localised DBpedia Datasets. *Proceedings of the 13th International Conference on Knowledge Management and Knowledge Technologies - i-Know '13*, p. 1–8, 2013.

APROSIO, A. P.; GIULIANO, C.; LAVELLI, A. Extending the coverage of DBpedia properties using distant supervision over Wikipedia. *CEUR Workshop Proceedings*, v. 1064, 2013. ISSN 16130073.

BAHARUDIN, B.; LEE, L. H.; KHAN, K. A Review of Machine Learning Algorithms for Text-Documents Classification. *Journal of Advances in Information Technology*, v. 1, n. 1, 2010. ISSN 1798-2340.

BALOG, K. *Entity-Oriented Search*. Cham: Springer International Publishing, 2018. (The Information Retrieval Series, v. 39). ISBN 978-3-319-93933-9.

BANERJEE, S.; TSIOUTSIOULIKLIS, K. Relation Extraction Using Multi-Encoder LSTM Network on a Distant Supervised Dataset. *Proceedings - 12th IEEE International Conference on Semantic Computing, ICSC 2018*, IEEE, v. 2018-Janua, p. 235–238, 2018.

BANKO, M.; CAFARELLA, M. J.; SODERLAND, S.; BROADHEAD, M.; ETZIONI, O. Open information extraction from the web. *IJCAI International Joint Conference on Artificial Intelligence*, v. 51, n. 12, p. 2670–2676, 12 2007. ISSN 10450823.

BLUNSOM, P.; COHN, T. Discriminative word alignment with conditional random fields. *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the ACL - ACL '06*, n. July, p. 65–72, 2006. ISSN 2194-9034.

BRIN, S. Extracting Patterns and Relations from the World Wide Web. *The World Wide Web and Databases*, v. 1590, n. 2, p. 172–183, 1999. ISSN 1098-6596.

BYRD, R. H.; LU, P.; NOCEDAL, J.; ZHU, C. A Limited Memory Algorithm for Bound Constrained Optimization. *SIAM Journal on Scientific Computing*, v. 16, n. 5, p. 1190–1208, 9 1995. ISSN 1064-8275.

CHANG, C.-c.; LIN, C.-j. LIBSVM. *ACM Transactions on Intelligent Systems and Technology*, v. 2, n. 3, p. 1–27, 4 2011. ISSN 21576904.

CHIU, J. P.; NICHOLS, E. Named Entity Recognition with Bidirectional LSTM-CNNs. *Transactions of the Association for Computational Linguistics*, Association for Computational Linguistics, Morristown, NJ, USA, v. 4, p. 357–370, 12 2016. ISSN 2307-387X.

COHEN, W.; RAVIKUMAR, P.; FIENBERG, S. A Comparison of String Distance Metrics for Name-Matching Task. In: *IJCAI 2003 Workshop on Information Integration on the Web (IIWeb 2003)*. [S.l.: s.n.], 2003. v. 47.

COLLOBERT, R.; WESTON, J.; BOTTOU, L.; KARLEN, M.; KAVUKCUOGLU, K.; KUKSA, P. Natural Language Processing (almost) from Scratch. v. 12, p. 2493–2537, 2011. ISSN 0891-2017.

CORTES, C.; VAPNIK, V. Support-Vector Networks. *Machine Learning*, v. 297, n. 20, p. 273–297, 1995. ISSN 1747-0285.

DEROSE, P.; SHEN, W.; CHEN, F.; LEE, Y. DBLife: A community information management platform for the database research community. *Cidr*, p. 1–4, 2007. ISSN 0361-803X.

DOAN, A.; HALEVY, A.; IVES, Z. *Principles of Data Integration*. 1st. ed. [S.l.]: Elsevier, 2012. 520 p. ISBN 9780124160446.

DONG, X.; GABRILOVICH, E.; HEITZ, G.; HORN, W.; LAO, N.; MURPHY, K.; STROHMANN, T.; SUN, S.; ZHANG, W. Knowledge vault: a web-scale approach to probabilistic knowledge fusion. *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '14*, p. 601–610, 2014. ISSN 0893-6080.

ELMAN, J. L. Finding Structure in Time. *Cognitive Science*, v. 14, n. 2, p. 179–211, 3 1990. ISSN 03640213.

FERRUCCI, D.; BROWN, E.; CHU-CARROLL, J.; FAN, J.; GONDEK, D.; KALYANPUR, A. A.; LALLY, A.; MURDOCK, J. W.; NYBERG, E.; PRAGER, J.; SCHLAEFER, N.; WELTY, C. Building Watson: An Overview of the DeepQA Project. *AI Magazine*, v. 31, n. 3, p. 59, 2010. ISSN 0738-4602.

FINKEL, J. R.; GRENAGER, T.; MANNING, C. Incorporating non-local information into information extraction systems by Gibbs sampling. *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics - ACL '05*, n. June, p. 363–370, 2005. ISSN 02773791.

FLEISCHHACKER, D.; PAULHEIM, H.; BRYL, V.; VÖLKER, J.; BIZER, C. Detecting errors in numerical linked data using cross-checked outlier detection. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, v. 8796, p. 357–372, 2014. ISSN 16113349.

FONT, L.; ZOUAQ, A.; GAGNON, M. Assessing and Improving Domain Knowledge Representation in DBpedia. *Open Journal of Semantic Web*, v. 4, n. 1, 2017.

GOLDBERG, Y. Neural Network Methods for Natural Language Processing (Synthesis Lectures on Human Language Technologies). *Synthesis Lectures on Human Language Technologies*, v. 10, n. 1, p. 1–309, 2017.

GRAVES, A.; SCHMIDHUBER, J. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural Networks*, v. 18, n. 5-6, p. 602–610, 7 2005. ISSN 08936080.

GRISHMAN, R. Information extraction: Techniques and challenges. In: . [S.l.]: Springer-Verlag, 1997. p. 10–27. ISBN 3-540-63438-X.

GUODONG, Z.; JIAN, S.; JIE, Z.; MIN, Z. Exploring various knowledge in relation extraction. *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics - ACL '05*, n. June, p. 427–434, 2005. ISSN 1650-1977.

HALEVY, A.; NORVIG, P.; PEREIRA, F. The Unreasonable Effectiveness of Data. *IEEE Intelligent Systems*, v. 24, n. 2, p. 8–12, 3 2009. ISSN 1541-1672.

HEARST, M. A. Automatic acquisition of hyponyms from large text corpora. In: *Proceedings of the 14th conference on Computational linguistics* -. Morristown, NJ, USA: Association for Computational Linguistics, 1992. v. 2, p. 539. ISSN 1469-8110.

HELLMANN, S.; LEHMANN, J.; AUER, S.; BRÜMMER, M. Integrating NLP Using Linked Data. In: *AIP Conference Proceedings*. [S.l.: s.n.], 2013. v. 1501, n. 1, p. 98–113. ISBN 978-3-642-41337-7.

HOCHREITER, S.; SCHMIDHUBER, J. Long Short-Term Memory. *Neural Computation*, v. 9, n. 8, p. 1735–1780, 11 1997. ISSN 0899-7667.

HOFFMANN, R.; ZHANG, C.; LING, X.; ZETTLEMOYER, L.; WELD, D. S. Knowledge-based weak supervision for information extraction of overlapping relations. *ACL-HLT 2011 - Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, v. 1, p. 541–550, 2011.

JARO, M. A. Advances in Record-Linkage Methodology as Applied to Matching the 1985 Census of Tampa, Florida. *Journal of the American Statistical Association*, Vol. 84, n. June 1989, No. 406, p. 414–420, 1989.

KRISHNAMURTHY, R.; LI, Y.; RAGHAVAN, S.; REISS, F.; VAITHYANATHAN, S.; ZHU, H. SystemT : A System for Declarative Information Extraction. v. 37, n. 4, p. 7–13, 2008.

LAFFERTY, J.; MCCALLUM, A.; PEREIRA, F. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data Part of the Numerical Analysis and Scientific Computing Commons Recommended Citation "Conditional Random Fields: Probabilistic Models for Segmenting and Labelin. *Proc. of ICML*, v. 2001, n. June, p. 282–289, 2001. ISSN 1750-2799.

LANGE, D.; BÖHM, C.; NAUMANN, F. *Extracting structured information from Wikipedia articles to populate infoboxes*. [S.l.: s.n.], 2010. 1661–1664 p. ISSN 16135652. ISBN 9781450300995.

LEHMANN, J.; ISELE, R.; JAKOB, M.; JENTZSCH, A.; KONTOKOSTAS, D.; MENDES, P. N.; HELLMANN, S.; MORSEY, M.; KLEEF, P. V.; AUER, S.; BIZER, C. DBpedia - A large-scale, multilingual knowledge base extracted from Wikipedia. *Semantic Web*, v. 6, n. 2, p. 167–195, 2015. ISSN 22104968.

LI, Y.; ROAD, H.; JOSE, S.; REISS, F. R.; ROAD, H.; JOSE, S.; CHITICARIU, L.; ROAD, H.; JOSE, S. SystemT : A Declarative Information Extraction System. n. June, p. 109–114, 2011.

MCCALLUM, A.; LI, W. Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons. In: *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003* -. Morristown, NJ, USA: Association for Computational Linguistics, 2003. v. 4, p. 188–191.

MIKOLOV, T.; SUTSKEVER, I.; CHEN, K.; CORRADO, G.; DEAN, J. Distributed Representations of Words and Phrases and their Compositionality. *CrossRef Listing of Deleted DOIs*, v. 1, p. 1–9, 10 2013. ISSN 0003-6951.

MINTZ, M.; BILLS, S.; SNOW, R.; JURAFSKY, D. Distant supervision for relation extraction without labeled data. In: *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2 - ACL-IJCNLP '09*. [S.l.: s.n.], 2009. v. 2, n. August, p. 1003. ISBN 9781932432466. ISSN 1932432469.

MOHAMED, M.; OUSSALAH, M. Identifying and Extracting Named Entities from Wikipedia Database Using Entity Infoboxes. *International Journal of Advanced Computer Science and Applications*, v. 5, n. 7, p. 164–169, 2014. ISSN 2158107X.

NGUYEN, H.; NGUYEN, T.; NGUYEN, H.; FREIRE, J. Querying Wikipedia documents and relationships. *Procceedings of the 13th International Workshop on the Web and Databases - WebDB '10*, p. 1, 2010. ISSN 07308078.

NGUYEN, T. H.; NGUYEN, H. D.; MOREIRA, V.; FREIRE, J. Clustering Wikipedia infoboxes to discover their types. *Proceedings of the 21st ACM international conference on Information and knowledge management - CIKM '12*, p. 2134, 2012.

NICKEL, M.; RING, O.-h. Factorizing YAGO Scalable Machine Learning for Linked Data. *Proc.\ of WWW'12*, p. 271–280, 2012. ISSN 1878-5832.

NIU, F.; ZHANG, C.; RÉ, C.; SHAVLIK, J. Elementary: Large-scale Knowledge-base Construction via Machine Learning and Statistical Inference. 2012.

PAULHEIM, H. Browsing Linked Open Data with Auto Complete. *11th International Semantic Web Conference (ISWC2012)*, p. 1–8, 2012. ISSN 16130073.

PAULHEIM, H. Identifying Wrong Links between Datasetsby Multi-dimensional Outlier Detection. In: *Third International Workshop on Debugging Ontologies and Ontology Mappings*. [S.l.: s.n.], 2014. p. 27–38. ISSN 1613-0073.

PAULHEIM, H. Knowledge graph refinement: A survey of approaches and evaluation methods. *Semantic Web*, v. 8, n. 3, p. 489–508, 12 2016. ISSN 22104968.

PAULHEIM, H. Data-Driven Joint Debugging of the DBpedia Mappings and Ontology. In: *The Semantic Web. ESWC 2017. Lecture Notes in Computer Science*. [S.l.]: Springer, Cham, 2017. p. 404–418.

PAULHEIM, H.; BIZER, C. Type Inference on Noisy RDF Data. In: . [S.l.: s.n.], 2013. p. 510–525. ISBN 978-3-642-41335-3.

PAULHEIM, H.; BIZER, C. Improving the Quality of Linked Data Using Statistical Distributions. *International Journal on Semantic Web and Information Systems*, v. 10, n. 2, p. 63–86, 2014. ISSN 1552-6283.

PENG, F.; MCCALLUM, A. Information extraction from research papers using conditional random fields. *Information Processing & Management*, v. 42, n. 4, p. 963–979, 7 2006. ISSN 03064573.

PENNINGTON, J.; SOCHER, R.; MANNING, C. Glove: Global Vectors for Word Representation. In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. [S.l.: s.n.], 2014. p. 1532–1543. ISBN 9781937284961. ISSN 10495258.

PINTO, D.; MCCALLUM, A.; WEI, X.; CROFT, W. B. Table extraction using conditional random fields. *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval - SIGIR '03*, p. 235, 2003. ISSN 01635840.

RIEDEL, S.; YAO, L.; MCCALLUM, A. Modeling Relations and Their Mentions without Labeled Text. In: *Proceedings of the 2010 European Conference on Machine Learning and Knowledge Discovery in Databases: Part III*. Berlin, Heidelberg: Springer-Verlag, 2010. p. 148–163. ISBN 3-642-15938-9, 978-3-642-15938-1.

RILOFF, E. Automatically Constructing a Dictionary for Information Extraction Tasks. *Proceedings of the 11th National Conference on Artificial Intelligence - AAAI '93*, p. 811–816, 1993. ISSN 1479-6678.

ROTH, B.; BARTH, T.; WIEGAND, M.; KLAKOW, D. A survey of noise reduction methods for distant supervision. In: *Proceedings of the 2013 Workshop on Automated Knowledge Base Construction*. New York, NY, USA: ACM, 2013. p. 73–77. ISBN 9781450324113.

SA ALEX RATNER, C. R. J. S. F. W. S. W. C. D.; ZHANG, C. DeepDive: Declarative Knowledge Base Construction. v. 143, n. 5, p. 951–959, 2017. ISSN 1527-5418.

SÁEZ, T.; HOGAN, A. Automatically Generating Wikipedia Info-boxes from Wikidata. *Companion of the The Web Conference 2018 on The Web Conference 2018 - WWW '18*, p. 1823–1830, 2018. Available at: <http://dl.acm.org/citation.cfm?doid=3184558.3191647>.

SANTOS, C. dos; XIANG, B.; ZHOU, B. Classifying Relations by Ranking with Convolutional Neural Networks. In: *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Stroudsburg, PA, USA: Association for Computational Linguistics, 2015. p. 626–634. ISBN 9781941643723. ISSN 9781937284435.

SCHUSTER, M.; PALIWAL, K. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, v. 45, n. 11, p. 2673–2681, 1997. ISSN 1053587X.

SHA, F.; PEREIRA, F. Shallow parsing with conditional random fields. *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - NAACL '03*, v. 1, n. June, p. 134–141, 2003. ISSN 0013-4686.

SHAPIRO, S. S.; WILK, M. B. An Analysis of Variance Test for Normality (Complete Samples). *Biometrika*, v. 52, n. 3/4, p. 591, 12 1965. ISSN 00063444.

SHEN, W.; DOAN, A.; NAUGHTON, J. F.; RAMAKRISHNAN, R. Declarative Information Extraction Using Datalog with Embedded Extraction Predicates. In: *VLDB '07 Proceedings of the 33rd international conference on Very large data bases*. Vienna, Austria: VLDB Endowment, 2007. p. 1033–1044. ISBN 978-1-59593-649-3.

SLEEMAN, J.; FININ, T. Type prediction for efficient coreference resolution in heterogeneous semantic graphs. *Proceedings - 2013 IEEE 7th International Conference on Semantic Computing, ICSC 2013*, p. 78–85, 2013.

SLEEMAN, J.; FININ, T.; JOSHI, A. Topic modeling for RDF graphs. *CEUR Workshop Proceedings*, v. 1467, p. 48–62, 2015. ISSN 16130073.

SODERLAND, S.; FISHER, D.; ASELTINE, J.; LEHNERT, W. CRYSTAL: Inducing a Conceptual Dictionary. *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, p. 1314–1321, 1995.

SUCHANEK, F. M.; KASNECI, G.; WEIKUM, G. Yago: A Core of Semantic Knowledge. *Proceedings of the 16th International Conference on World Wide Web*, p. 697–706, 2007. ISSN 01695347. Available at: <http://dl.acm.org/citation.cfm?id=1242572.1242667>.

SURDEANU, M.; TIBSHIRANI, J.; NALLAPATI, R.; MANNING, C. D. Multi-instance Multi-label Learning for Relation Extraction. n. July, p. 455–465, 2010.

SUTTON, C. An Introduction to Conditional Random Fields. *Foundations and Trends® in Machine Learning*, v. 4, n. 4, p. 267–373, 2012. ISSN 1935-8237.

TAKAMATSU, S.; SATO, I.; NAKAGAWA, H. Reducing Wrong Labels in Distant Supervision for Relation Extraction. *Jeju, Republic of Korea*, n. July, p. 721–729, 2012.

VO, D. T.; BAGHERI, E. Open Information Extraction. v. 1, n. C, p. 11p., 2016.

WELD, D. S.; HOFFMANN, R. Using Wikipedia to Bootstrap Open Information Extraction. v. 37, n. 4, p. 62–68, 2008.

WIENAND, D.; PAULHEIM, H. Detecting incorrect numerical data in DBpedia. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, v. 8465 LNCS, p. 504–518, 2014. ISSN 16113349.

WILCOXON, F. Individual Comparisons by Ranking Methods. *Biometrics Bulletin*, v. 1, n. 6, p. 80, 12 1945. ISSN 00994987.

WINKLER, W. E. The State of Record Linkage and Current Research Problems. *US Census Bureau*, p. 1–15, 1999. ISSN 0031-6768.

WU, F.; WELD, D. S. Autonomously semantifying wikipedia. *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management - CIKM '07*, p. 41, 2007. ISSN 9781595938039.

XING, Z.; PEI, J.; KEOGH, E. A brief survey on sequence classification. *ACM SIGKDD Explorations Newsletter*, v. 12, n. 1, p. 40, 11 2010. ISSN 19310145.

YOUNG, T.; HAZARIKA, D.; PORIA, S.; CAMBRIA, E. Recent trends in deep learning based natural language processing [Review Article]. *IEEE Computational Intelligence Magazine*, v. 13, n. 3, p. 55–75, 2018. ISSN 15566048.

YUS, R.; MULWAD, V.; FININ, T.; MENA, E. Infoboxer: Using statistical and semantic knowledge to help create Wikipedia infoboxes. *CEUR Workshop Proceedings*, v. 1272, p. 405–408, 2014. ISSN 16130073.

ZAVERI, A.; KONTOKOSTAS, D.; SHERIF, M. A.; BÜHMANN, L.; MORSEY, M.; AUER, S.; LEHMANN, J. User-driven quality evaluation of DBpedia. *Proceedings of the 9th International Conference on Semantic Systems - I-SEMANTICS '13*, p. 97, 2013. ISSN 9781450319720.

ZHANG, X.; ZHAO, J.; LECUN, Y. Character-level Convolutional Networks for Text Classification. *NIPS'15 Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, v. 1, 9 2015. ISSN 17429552.

ZHANG, Y.; WALLACE, B. A Sensitivity Analysis of (and Practitioners' Guide to) Convolutional Neural Networks for Sentence Classification. p. 253–263, 2015. ISSN 10709908.

ZHOU, G.; SU, J. Named entity recognition using an HMM-based chunk tagger. In: *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics - ACL '02*. Morristown, NJ, USA: Association for Computational Linguistics, 2001. p. 473.

**Source:** This research

| INFOBOX TEMPLATE | MODEL | MACRO | | | MICRO | | |
|---|---|---|---|---|---|---|---|
| | | PREC. | RECALL | F-SCORE | PREC. | RECALL | F-SCORE |
| US County | DEEPEX-BLSTM | 0.6672 | 1.0000 | 0.8004 | 0.7050 | 1.0000 | 0.8270 |
| | DEEPEX-BLSTM_W2 | 0.6685 | 0.9722 | 0.7923 | 0.7068 | 0.9975 | 0.8274 |
| | DEEPEX-BLSTM+CNN | 0.7147 | 0.9698 | **0.8230** | 0.7568 | 0.9953 | **0.8598** |
| | DEEPEX-CRF | 0.6846 | 0.9942 | 0.8109 | 0.7196 | 0.9926 | 0.8344 |
| | iPopulator | 0.4462 | 0.6611 | 0.5328 | 0.5105 | 0.7246 | 0.5990 |
| | Kylin | 0.5264 | 0.4859 | 0.5053 | 0.8218 | 0.5425 | 0.6535 |
| Airline | DEEPEX-BLSTM | 0.2874 | 0.5786 | 0.3840 | 0.4500 | 0.7500 | 0.5625 |
| | DEEPEX-BLSTM_W2 | 0.3313 | 0.5912 | **0.4246** | 0.4844 | 0.7561 | **0.5905** |
| | DEEPEX-BLSTM+CNN | 0.2738 | 0.4593 | 0.3431 | 0.4227 | 0.7069 | 0.5290 |
| | DEEPEX-CRF | 0.3242 | 0.4503 | 0.3770 | 0.5092 | 0.6917 | 0.5866 |
| | iPopulator | 0.2541 | 0.3580 | 0.2972 | 0.4636 | 0.5426 | 0.5000 |
| | Kylin | 0.1993 | 0.4655 | 0.2791 | 0.4545 | 0.4762 | 0.4651 |
| Artist | DEEPEX-BLSTM | 0.6332 | 0.9713 | 0.7666 | 0.6296 | 0.9684 | 0.7631 |
| | DEEPEX-BLSTM_W2 | 0.6038 | 0.9795 | 0.7471 | 0.6025 | 0.9800 | 0.7462 |
| | DEEPEX-BLSTM+CNN | 0.6326 | 0.9695 | 0.7656 | 0.6307 | 0.9682 | 0.7638 |
| | DEEPEX-CRF | 0.6932 | 0.9482 | **0.8009** | 0.6920 | 0.9480 | **0.8000** |
| | iPopulator | 0.3088 | 0.3422 | 0.3247 | 0.3407 | 0.4946 | 0.4035 |
| | Kylin | 0.2745 | 0.4000 | 0.3256 | 0.6800 | 0.3417 | 0.4548 |
| University | DEEPEX-BLSTM | 0.6803 | 0.9168 | 0.7811 | 0.6910 | 0.9360 | 0.7951 |
| | DEEPEX-BLSTM_W2 | 0.6854 | 0.9231 | **0.7867** | 0.6966 | 0.9422 | **0.8010** |
| | DEEPEX-BLSTM+CNN | 0.6621 | 0.9541 | 0.7818 | 0.6681 | 0.9636 | 0.7891 |
| | DEEPEX-CRF | 0.6824 | 0.7962 | 0.7349 | 0.6814 | 0.8035 | 0.7374 |
| | iPopulator | 0.3146 | 0.5152 | 0.3907 | 0.3295 | 0.5758 | 0.4191 |
| | Kylin | 0.4383 | 0.4093 | 0.4233 | 0.7333 | 0.3850 | 0.5049 |

Table 14 – Complete Macro and Micro results from evaluation experiments.

**Source:** This research

| INFOBOX TEMPLATE | DATASET | DEEPEX | | | | BASELINES | |
|---|---|---|---|---|---|---|---|
| | | BLSTM | BLSTM_W2 | BLSTM+CNN | CRF | IPOPULATOR | KYLIN |
| **US County** | area_land_sq_mi | **1.00000** | **1.00000** | **1.00000** | 0.80000 | 0.98000 | 0.98000 |
| | area_percentage | 0.86000 | 0.88000 | 0.78000 | **0.98000** | 0.28571 | 0.88000 |
| | area_total_sq_mi | **1.00000** | **1.00000** | **1.00000** | 0.98000 | 0.98000 | 0.96000 |
| | area_water_sq_mi | 0.66000 | 0.64000 | 0.70000 | 0.66000 | **0.97959** | 0.94000 |
| | county | **1.00000** | **1.00000** | **1.00000** | **1.00000** | 0.95652 | 0.00000 |
| | density_sq_mi | 0.32000 | 0.30000 | 0.44000 | 0.42000 | 0.06000 | **0.60714** |
| | district | 0.10000 | 0.10526 | 0.12500 | 0.06897 | 0.00000 | 0.00000 |
| | largest_city_wl | 0.22500 | 0.21951 | 0.20930 | 0.23529 | **0.30000** | 0.00000 |
| | named_for | **0.76087** | 0.71739 | 0.72340 | 0.48000 | 0.43902 | 0.52000 |
| | pop | 0.38000 | 0.38000 | **0.84000** | 0.78000 | 0.02041 | 0.00000 |
| | seat_wl | 0.90000 | **0.96000** | 0.95918 | 0.85106 | 0.15909 | 0.84615 |
| | state | 0.80000 | 0.82000 | 0.80000 | **0.96000** | 0.19355 | 0.58333 |
| **Airline** | airline | 0.78000 | 0.76000 | 0.71429 | **0.95349** | 0.58537 | 0.73469 |
| | callsign | 0.20000 | **0.40000** | 0.12500 | 0.33333 | 0.00000 | 0.09091 |
| | fleet_size | 0.00000 | 0.00000 | 0.00000 | 0.00000 | 0.00000 | 0.00000 |
| | founded | 0.61224 | **0.66667** | 0.63830 | 0.37209 | 0.05556 | 0.44444 |
| | headquarters | 0.39583 | 0.46809 | 0.40541 | **0.51064** | 0.00000 | 0.00000 |
| | IATA | 0.02381 | 0.02439 | 0.03333 | 0.10000 | 0.00000 | **0.12500** |
| | ICAO | 0.00000 | 0.00000 | 0.00000 | 0.00000 | 0.00000 | 0.00000 |
| **Artist** | birth_date | **0.93750** | 0.91667 | 0.89583 | 0.83333 | 0.00000 | 0.63265 |
| | birth_place | 0.36000 | 0.38776 | 0.36735 | **0.56818** | 0.03448 | 0.00000 |
| | field | 0.52000 | 0.50000 | 0.51020 | **0.54000** | 0.22581 | 0.00000 |
| | name | 0.62500 | 0.52083 | 0.69388 | **0.82000** | 0.45946 | 0.74000 |
| | nationality | 0.72340 | 0.69388 | 0.69565 | **0.76190** | 0.25000 | 0.00000 |
| **University** | city | 0.51220 | 0.51163 | 0.48936 | **0.62963** | 0.10000 | 0.00000 |
| | country | 0.75510 | 0.71429 | 0.68750 | **0.85294** | 0.05556 | 0.00000 |
| | established | 0.75000 | **0.79592** | 0.69388 | 0.68750 | 0.46154 | 0.62500 |
| | name | 0.94000 | 0.94000 | 0.94000 | **0.95918** | 0.57143 | 0.90000 |
| | type | 0.44444 | 0.46512 | 0.50000 | 0.28261 | 0.38462 | **0.66667** |

Table 15 – Complete precision measures for each dataset.

**Source:** This research

| INFOBOX TEMPLATE | DATASET | DEEPEX | | | | BASELINES | |
|---|---|---|---|---|---|---|---|
| | | BLSTM | BLSTM_W2 | BLSTM+CNN | CRF | IPOPULATOR | KYLIN |
| **US County** | area_land_sq_mi | **1.00000** | **1.00000** | **1.00000** | **1.00000** | **1.00000** | **1.00000** |
| | area_percentage | **1.00000** | **1.00000** | **1.00000** | **1.00000** | 0.50000 | **1.00000** |
| | area_total_sq_mi | **1.00000** | **1.00000** | **1.00000** | **1.00000** | **1.00000** | **1.00000** |
| | area_water_sq_mi | **1.00000** | **1.00000** | **1.00000** | **1.00000** | 0.88889 | **1.00000** |
| | county | **1.00000** | **1.00000** | **1.00000** | **1.00000** | 0.51765 | 0.00000 |
| | density_sq_mi | **1.00000** | **1.00000** | **1.00000** | **1.00000** | **1.00000** | 0.43590 |
| | district | **1.00000** | 0.66667 | 0.66667 | **1.00000** | 0.00000 | 0.00000 |
| | largest_city_wl | **1.00000** | **1.00000** | **1.00000** | **1.00000** | **1.00000** | 0.00000 |
| | named_for | **1.00000** | **1.00000** | 0.97143 | **1.00000** | 0.51429 | **1.00000** |
| | pop | **1.00000** | **1.00000** | **1.00000** | **1.00000** | 0.50000 | 0.00000 |
| | seat_wl | **1.00000** | **1.00000** | **1.00000** | 0.93023 | 0.46667 | 0.23913 |
| | state | **1.00000** | **1.00000** | **1.00000** | **1.00000** | 0.54545 | 0.15556 |
| **Airline** | airline | **1.00000** | **1.00000** | 0.97222 | 0.85417 | 0.51064 | 0.97297 |
| | callsign | 0.10000 | 0.18182 | 0.09091 | 0.10000 | 0.00000 | 0.28571 |
| | fleet_size | 0.00000 | 0.00000 | 0.00000 | 0.00000 | 0.00000 | 0.00000 |
| | founded | **1.00000** | **1.00000** | **1.00000** | 0.94118 | 0.50000 | **1.00000** |
| | headquarters | 0.95000 | 0.95652 | 0.65217 | 0.92308 | 0.00000 | 0.00000 |
| | IATA | **1.00000** | **1.00000** | 0.50000 | 0.33333 | 0.00000 | **1.00000** |
| | ICAO | 0.00000 | 0.00000 | 0.00000 | 0.00000 | 0.00000 | 0.00000 |
| **Artist** | birth_date | **1.00000** | **1.00000** | **1.00000** | 0.97561 | 0.00000 | **1.00000** |
| | birth_place | **1.00000** | **1.00000** | **1.00000** | 0.92593 | 0.12500 | 0.00000 |
| | field | **1.00000** | **1.00000** | 0.96154 | **1.00000** | 0.38889 | 0.00000 |
| | name | 0.93750 | 0.92593 | 0.97143 | **1.00000** | 0.56667 | **1.00000** |
| | nationality | 0.91892 | **0.97143** | 0.91429 | 0.84211 | 0.33333 | 0.00000 |
| **University** | city | 0.77778 | 0.84615 | **0.92000** | 0.51515 | 0.50000 | 0.00000 |
| | country | **1.00000** | **1.00000** | 0.97059 | 0.67442 | 0.33333 | 0.00000 |
| | established | 0.97297 | **1.00000** | **1.00000** | **1.00000** | 0.60000 | **1.00000** |
| | name | **1.00000** | **1.00000** | **1.00000** | 0.97917 | 0.66667 | **1.00000** |
| | type | 0.83333 | 0.76923 | **0.88000** | 0.81250 | 0.47619 | 0.04651 |

Table 16 – Complete recall measures for each dataset.

**Source:** This research

| INFOBOX TEMPLATE | DATASET | DEEPEX | | | | BASELINES | |
|---|---|---|---|---|---|---|---|
| | | BLSTM | BLSTM_W2 | BLSTM+CNN | CRF | IPOPULATOR | KYLIN |
| **U.S. County** | area_land_sq_mi | **1.00000** | **1.00000** | **1.00000** | 0.88889 | 0.98990 | 0.98990 |
| | area_percentage | 0.92473 | 0.93617 | 0.87640 | **0.98990** | 0.36364 | 0.93617 |
| | area_total_sq_mi | **1.00000** | **1.00000** | **1.00000** | 0.98990 | 0.98990 | 0.97959 |
| | area_water_sq_mi | 0.79518 | 0.78049 | 0.82353 | 0.79518 | 0.93204 | **0.96907** |
| | county | **1.00000** | **1.00000** | **1.00000** | **1.00000** | 0.67176 | 0.00000 |
| | density_sq_mi | 0.48485 | 0.46154 | **0.61111** | 0.59155 | 0.11321 | 0.50746 |
| | district | 0.18182 | 0.18182 | **0.21053** | 0.12903 | 0.00000 | 0.00000 |
| | largest_city_wl | 0.36735 | 0.36000 | 0.34615 | 0.38095 | **0.46154** | 0.00000 |
| | named_for | **0.86420** | 0.83544 | 0.82927 | 0.64865 | 0.47368 | 0.68421 |
| | pop | 0.55072 | 0.55072 | **0.91304** | 0.87640 | 0.03922 | 0.00000 |
| | seat_wl | 0.94737 | **0.97959** | 0.97917 | 0.88889 | 0.23729 | 0.37288 |
| | state | 0.88889 | 0.90110 | 0.88889 | **0.97959** | 0.28571 | 0.24561 |
| **Airline** | airline | 0.87640 | 0.86364 | 0.82353 | **0.90110** | 0.58586 | 0.83721 |
| | callsign | 0.13333 | 0.25000 | 0.10526 | 0.15385 | **0.28571** | 0.13793 |
| | fleet_size | 0.00000 | 0.00000 | 0.00000 | 0.00000 | **0.36364** | 0.00000 |
| | founded | 0.75949 | **0.80000** | 0.77922 | 0.53333 | 0.58621 | 0.61538 |
| | headquarters | 0.55882 | 0.62857 | 0.50000 | **0.65753** | 0.14286 | 0.22222 |
| | IATA | 0.04651 | 0.04762 | 0.06250 | **0.15385** | 0.00000 | 0.00000 |
| | ICAO | 0.00000 | 0.00000 | 0.00000 | 0.00000 | 0.00000 | 0.00000 |
| **Artist** | birth_date | **0.96774** | 0.95652 | 0.94505 | 0.89888 | 0.04444 | 0.77500 |
| | birth_place | 0.52941 | 0.55882 | 0.53731 | **0.65753** | 0.05405 | 0.00000 |
| | field | 0.68421 | 0.66667 | 0.66667 | **0.70130** | 0.44068 | 0.00000 |
| | name | 0.75000 | 0.66667 | 0.80952 | **0.90110** | 0.75000 | 0.85057 |
| | nationality | **0.80952** | 0.80952 | 0.79012 | 0.80000 | 0.28571 | 0.00000 |
| **University** | city | 0.61765 | 0.63768 | **0.63889** | 0.56667 | 0.16667 | 0.00000 |
| | country | **0.86047** | 0.83333 | 0.80488 | 0.75325 | 0.09524 | 0.00000 |
| | established | 0.84706 | **0.88636** | 0.81928 | 0.81481 | 0.52174 | 0.76923 |
| | name | **0.96907** | **0.96907** | **0.96907** | **0.96907** | 0.61538 | 0.94737 |
| | type | 0.57971 | 0.57971 | **0.63768** | 0.41935 | 0.42553 | 0.08696 |

Table 17 – Complete F-score measures for each dataset.