Alma Mater Studiorum · Università di Bologna

DIPARTIMENTO DI INTERPRETAZIONE E TRADUZIONE
Corso di Laurea magistrale in Specialized Translation (classe LM-94)

TESI DI LAUREA
in COMPUTATIONAL LINGUISTICS

Methods of Definition Extraction and Linking for Food Recipes

CANDIDATA:
Margherita Martinelli

RELATORE:
Alberto Barrón Cedeño

CORRELATORI:
Adriano Ferraresi
Federico Garcea

Anno Accademico 2020/2021
Terzo Appello

Acknowledgements

I would like to thank professor Alberto Barrón Cedeño, professor Adriano Ferraresi, professor Silvia Bernardini and professor Federico Garcea for their guidance and support during this research project.

I would also like to thank my parents for giving me the opportunity to study all these years, with all the support I needed and, most importantly, for believing I could make it.

Abstract

This dissertation deals with definitional contexts extraction and automatic definitions linking in the Italian and English language. Definitional contexts extraction is a task that is not limited to glossaries and encyclopaediae, but has been addressed also in the field of Natural Language Processing. In this research, the objective is to identify definitional contexts in food-related Wikipedia articles. To set the basis of the work, we built two ad-hoc corpora out of the Italian and English dumps of Wikipedia. We trained two BERT models in a supervised fashion with a manually annotated dataset. The F_1-measures of 96.08 and 97.66 testify the high performance. We then fed each model with 30 Wikipedia articles randomly extracted from the two corpora, one with Italian and one with English articles. We obtained the best results by restricting the selection to the first sentence of the article whose BERT positive score is above 0.6.

The task of automatic definitions linking is loosely based on the *wikification* process. Rather than linking a term to its corresponding Wikipedia article, we aim at linking a term to its corresponding definition in a Wikipedia article. To lay the foundation of the task, we built two ad-hoc corpora from a cooking website in its Italian and English version. We created a pipeline for automatic definitions linking and carried out a successful experiment using the title of a recipe as input text, the output of which is a minimalistic HTML version of the input, whose terms are linked to their corresponding Wikipedia articles. The definitions linking is one of the two missing steps in the pipeline and discussed in the conclusions.

The main contributions of this work are: building of four ad-hoc corpora, two from a cooking website in its Italian and English version and two from the Italian and English dumps of Wikipedia; training of two BERT models with a manually annotated dataset for the definitional contexts extraction; creation of a pipeline for the automatic definitions linking.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This research work has two aims. The first aim is to create a supervised model for the automatic identification of definitions and its subsequent application as a definition extractor given a document that belongs to the food domain. The second aim is the implementation of a pipeline for a system that produces a minimalistic HTML version of a recipe with the keywords linked to their corresponding definitions extracted from Wikipedia articles that belong to the food domain. The working languages are Italian and English. Therefore, this dissertation can be viewed as a two-sided coin: on one side, it deals with definitional context extraction; on the other side, it deals with automatic definitions linking, a process that follows the basic concepts of what is known as *wikification*.

These two tasks are fulfilled by exploiting computational linguistics concepts and techniques using the programming language Python[1].

To set the basis of the work, we built four ad-hoc corpora: two contain Italian and English recipes extracted from a cooking website; two contain food-related Wikipedia articles extracted from the Italian and English dumps of Wikipedia. To achieve the first goal, we trained two BERT models with the same manually annotated dataset. The high performance is confirmed by the F_1-measures of 96.08 and 97.66 and accuracy scores of 96.82 and 98.09. To investigate their performance on an unseen dataset, we fed each model 30 Wikipedia articles randomly selected from the two corpora, one batch in English and the other in Italian. Since the goal of the models is to extract the corresponding definitions of a term, we took into consideration

[1]The code is available at `https://github.com/TinfFoil/defXlink`.

two approaches: “top score” and “pos”. The first one refers to the segment that has the highest BERT score in the document; the second one refers to the first segment in the article whose BERT score is higher than 0.6. A manual evaluation proved that, for both approaches, the segments extracted from the articles are definitions, so the models technically work in the majority of the cases. But since the aim of this research is to extract the exact corresponding definitional context per term (and therefore discard other candidates), we concluded that the “pos” approach worked best compared to the “top score” by a small margin and for both models (i.e. both for Italian and English Wikipedia articles).

The second goal of this work was to create a pipeline for the automatic definitions linking. This was partially achieved by the results obtained from the first goal and the development of an experimental system that functions as follows:

1. When given an input text (the title of a recipe in our case), it detects the language;

2. It produces a list of n-grams;

3. It checks if there is a corresponding Wikipedia article for each n-gram;

4. It produces a minimalistic HTML version of the input text, pairing each n-gram with the corresponding article;

5. It assigns the class “other” or “food” according to the type of n-gram;

6. It extracts the definitional context in the existing Wikipedia article and assigns it to the corresponding n-gram;

7. It links the the n-gram to its definition.

This aim was partially achieved and the missing steps are two, namely the detection of the language of the input text and the linking of the definitions to the corresponding n-grams. The absence of these stages is discussed and proposed, together with other two departing points, as a future work in the conclusive Chapter of the dissertation.

1.1 General and Specific Objectives

The general objectives are two, definitional context identification and automatic definitions linking. In order to achieve these objectives, other two objectives had to be achieved as basis for the development of the two main ones. Each of these tasks are carried out both for the Italian and English language.

1. Recipes data acquisition:
   (a) Scrape a cooking website using `BeautifulSoup` library in order to collect recipes;
   (b) Clean the recipes;
   (c) For each recipe, create a folder containing four cleaned TXT files and the images associated to the descriptions of the steps.
2. Wikipedia data acquisition:
   (a) Parse the dumps of Wikipedia extracting only pages whose categories belong to the food domain;
   (b) Clean articles;
   (c) For each Wikipedia article, create a TXT file.
3. Definitional contexts extraction:
   (a) Train two BERT models with a labelled dataset;
   (b) Feed the two BERT models with real documents to assess their performance as definition extractors;
   (c) Manually evaluate the extracted definitions.
4. Automatic definitions linking:
   (a) Create the pipeline of a system that, when fed with an input text, it outputs a minimalistic HTML version of the text and links the n-grams to their corresponding definitional contexts extracted from Wikipedia articles.

1.2 Dissertation Structure

This dissertation is composed of five chapters.

Chapter 1 is the Introduction. It presents the research work and its objectives.

Chapter 2 is the Background. Firstly, it introduces the notion of Natural Language Processing and dives deeper into the description of the type of models used for this project. Secondly, it presents the data acquisition process. Thirdly, it delineates the theoretical and practical aspects of the definitional contexts identification process. Finally, it illustrates the *wikification* task and the systems developed to address it, which served as groundwork for the creation of a pipeline modified according to our needs.

Chapter 3 analyses the Definitional Contexts Identification task. It introduces the two corpora built from the Italian and English dumps of Wikipedia. It explains the metrics used to evaluate the models performance. It introduces the two models chosen for the task. After a thorough explanation of the experiments, it examines the results. Finally, it tests the models with real documents and analyses the output.

Chapter 4 proposes a pipeline for the Automatic Definition Linking task. It presents the two corpora built from the Italian and English versions of a cooking website and it describes the automatic definition linking process.

Chapter 5 draws the conclusions by summarizing what we have achieved so far and proposes three departing points for potential future works.

Chapter 2

Background

This chapter delineates the core concepts necessary to understand this research work. Section 2.1 gives a definition of natural language processing and presents the Transformer models. Section 2.2 explains the notion of web scraping, introducing the libraries used to parse the required data. Section 2.3 first analyses the definitional context and then discusses the related work on the topic, focusing on one model against which our results will be compared. Section 2.4 defines *wikification* and gives a panoramic of four systems whose concepts function as baseline for the second aim of this work.

2.1 Natural Language Processing

This research work relies on principles and techniques of natural language processing (NLP), which is defined as follows:

> Natural language processing is an area of research in computer science and artificial intelligence (AI) concerned with processing natural languages such as English or Mandarin. This processing generally involves translating natural language into data (numbers) that a computer can use to learn about the world. And this understanding of the world is sometimes used to generate natural language text that reflects that understanding (Hobson et al., 2019).

NLP normally follows either one of two approaches: rule-based and statistical. In the first and oldest approach the algorithm (or model) is composed by human-defined linguistic rules, hence the name rule-based. In the second approach the model is trained on a set of documents, either labeled or unlabeled, and learns to create its own rules for the task at hand (Hobson et al., 2019).

Among the variety of statistical models used in this field, we will cover the essential types useful to understand the task at hand: unsupervised, supervised and self-supervised. Unsupervised models learn directly from unlabelled data, meaning they do not require any sort of human-labeled inputs. Supervised models need labelled data to learn a task and their improvement is based upon the difference between the predictions and the expected output (Hobson et al., 2019). They are used for tasks like image classification and regression. Self-supervised models are fed with unlabelled data (just like the unsupervised ones) but tackle tasks traditionally targeted by supervised learning.

The field of machine learning has seen major improvements with the rise of statistical models that detach themselves from the constraints of the rule-based approach by developing Artificial Neural Networks (ANNs), or simply Neural Networks (NNs). A NN is an interconnected group of nodes (referred to as neurons) that process the information in a distributed fashion to learn from the input so that the final output can be optimized. Figure 2.1 shows a representation of a feed-forward neural network. Each circle on the left-hand side represents an input unit (x_0, x_1, x_2 and x_3) that the model receives (or is fed with). Each unit is turned into a vector representation of the input that the neural network can process. The inputs are then passed from the input layer to multiple hidden layers in a feed-forward fashion, where the input moves always in one direction and never goes backwards. The final layer, called output later, produces the output (y_1 and y_2) on the right-hand side.

Figure 2.2 zooms into a single neuron. Each input vector (x_1, x_2, x_3) is assigned a set of weights (w_1, w_2, w_3), which will be multiplied and summed together to obtain a single value. The neuron has a threshold called activation function that indicates whether it can fire or not: if the weighted sum surpasses that threshold, the neuron outputs 1, if it is lower, it outputs 0. The bias is an "always on" input to the neuron (Hobson et al., 2019), an additional parameter used to adjust the output of the model. Therefore: $output = sum(weights \cdot inputs) + bias$.

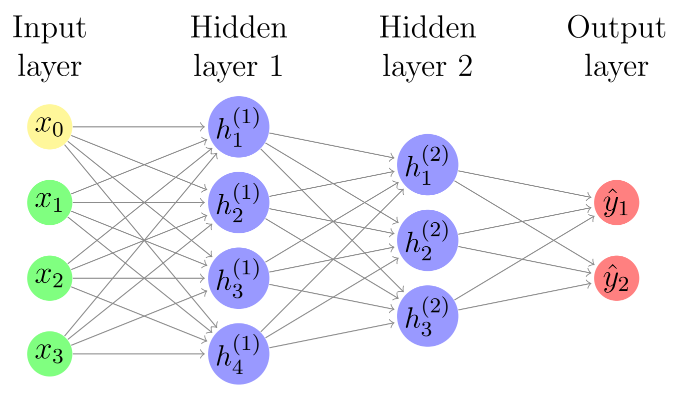


Figure 2.1: A neural network.

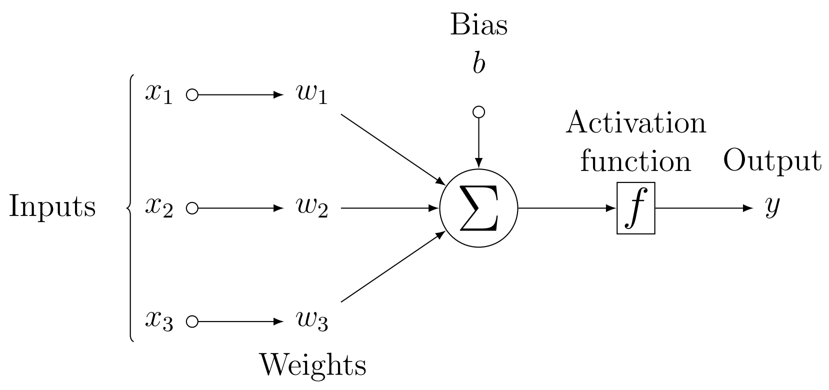


Figure 2.2: A close-up view of a neuron.

2.1.1 Transformer Models

Nowadays, Transformers (Vaswani et al., 2017) are the most popular neural networks in the field of NLP. First introduced in 2017, they were pre-trained on huge amounts of raw text in a self-supervised fashion to develop a statistical understanding of language. One of their main characteristics is the self-attention mechanism that mimics cognitive attention and allows the models to rely on long-term memory, both to "remember" the previous tokens in the sequence and to pay particular attention to the most relevant ones in the context. Transformers were originally developed for translation tasks (Vaswani et al., 2017) and can be divided into three types: encoders, decoders and encoder-decoders. Here we will focus on the first one.

The encoder model is a stack of encoder blocks. When given an input sequence $X_{1:n}$, for example *"I want to buy a car"*, each token is turned into an input vector $x'_j, \forall j \in \{1, ..., n\}$ and put into a dependency relation with all other input vectors $x'_1, ..., x'_n$. Therefore, in each encoder block, each input vector is turned from a context-independent vector representation (of a token) into a context-dependent vector representation. This operation is carried out in multiple encoder blocks, so that the dependency in the contextual representation is further refined until the last encoder block outputs the final contextual encoding. This process is better visualized in Figure 2.3. On the left-hand side, each input vector fed to the encoder blocks represents a token in the sequence (EOS stands for "End Of Sentence" and is the final input vector that prompts the model to finish the sequence); on the right-hand side, a closer look at the second block shows the dependency of each vector to all the other vectors in the sequence. Their pretraining usually consists in feeding them a corrupted sentence (i.e. a sentence with missing words) with the objective to reconstruct it into the original version. The most widely used encoder models derive from BERT (Devlin et al., 2018) and are mostly used for fine-tuning on a downstream task. Their training for fine-tuning, called transfer learning, is done in a supervised fashion by feeding them with human-annotated data. Some common tasks for which they are used are sentence classification (Madabushi et al., 2020), named entity recognition (Souza et al., 2019; Hakala and Pyysalo, 2019) and extracting question answering (Qu et al., 2019).

Nowadays there are many types of Transformers freely available from the Transformers library (Wolf et al., 2020)[1]. Here, we will focus on two encoder models, both deriving from BERT (Devlin et al., 2018): `bert-base-cased` and `bert-base-multilingual-cased`. They will be presented in Section 3.3.

2.2 Data Acquisition

Data acquisition from the Web is a very common practice nowadays and can be performed with many techniques. For the purpose of this research work, we will focus on web scraping. Also known as web extraction or web harvesting, it is a technique used to extract data from the World Wide Web

[1]For more information see `https://huggingface.co/models`; last visit 21.01.2022

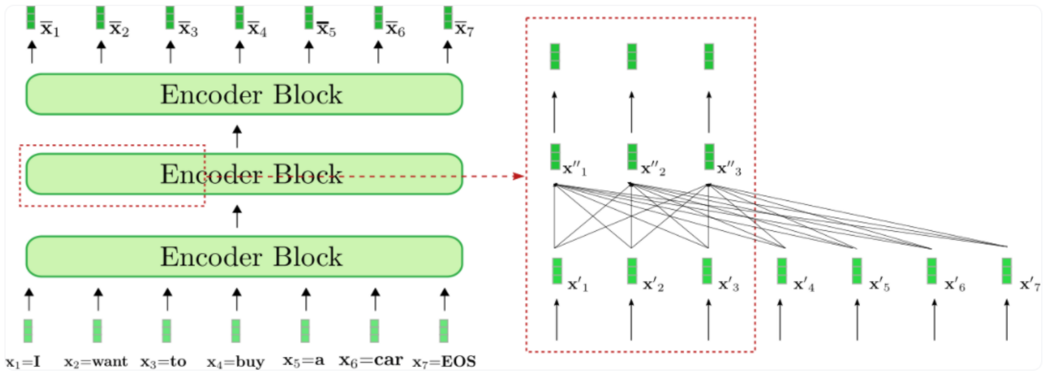

$\bar{x}_1$ $\bar{x}_2$ $\bar{x}_3$ $\bar{x}_4$ $\bar{x}_5$ $\bar{x}_6$ $\bar{x}_7$
Encoder Block
Encoder Block
Encoder Block
x_1=I x_2=want x_3=to x_4=buy x_5=a x_6=car x_7=EOS
x''_1 x''_2 x''_3
x'_1 x'_2 x'_3 x'_4 x'_5 x'_6 x'_7

Figure 2.3: Transformer-based encoder model, borrowed from `https://huggingface.co/blog/encoder-decoder`.

and store it to a file system or database for later exploitation or analysis (Zhao, 2017).

For this research work, our target were two sources: a website in its Italian and English version (the process is further described in section 4.1) and the Italian and English dumps of Wikipedia (analysed in section 3.1).

To build the two corpora of Italian and English recipes, we followed the full pipeline of the web scraping process, as described by Persson (2019), which consists of three stages, also shown in Figure 2.4:

1. Fetching stage: the target website is accessed via the HTTP protocol that sends requests from web servers to get the HTML page;

2. Extraction stage: once the HTML page is retrieved, the data contained in it is extracted by exploiting techniques such as regular expressions and HTML parsing libraries;

3. Transformation stage: the data is converted into a structured format for storage.

For the first stage, we used the `requests` library[2]. For the second stage, we used a Python package called `BeautifulSoup` (Richardson, 2007)[3]. It

[2]Available at `https://docs.python-requests.org/en/master/user/quickstart/`; last visit 23.02.2022

[3]`https://www.crummy.com/software/BeautifulSoup/bs4/doc/;lastvisit:31.01.2022`

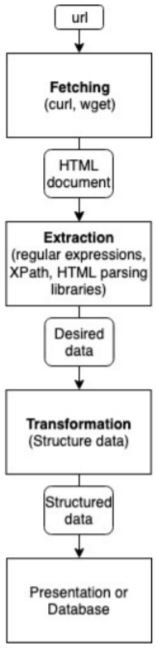


Figure 2.4: Web scraping process (Persson, 2019).

allows the user to retrieve structured data from a webpage by parsing its HTML or XML (Khder, 2021). For the third stage, each recipe was cleaned and stored into a folder containing four txt files (title, presentation, ingredients and preparation) and the images of the preparation. For a thorough explanation of the storage, see Appendix A.

To build the two corpora from the Italian and English Wikipedia dumps, the first stage was not necessary, since the dumps are readily available for download in wikicode, which is XML format. The second stage was performed using `mwparserfromhell`[4], a library that provides a parser for wikicode, which is in XML format. For the third stage, each Wikipedia article was cleaned and stored into a txt file. To have a full description of the storage, see Appendix A.

We will use both sources to build four ad-hoc corpora, two derived from the website and two derived from Wikipedia. The first two are explained in Chapter 4; the latter two in Chapter 3.

[4]`https://mwparserfromhell.readthedocs.io/en/latest/`; last visit: 31.01.2022

$$\underbrace{Gnocchi}_{definiendum}\ \underbrace{are}_{definitor}\ \underbrace{a\ varied\ family\ of\ dumplings}_{definiens}\underbrace{in\ the\ Italian\ cuisine}_{differentiae\ specificae}$$

Figure 2.5: An example of definitional context.

2.3 Definitional Contexts Identification

A definitional context is the explanation of a term. The philosopher Aristotle formulated the concept as follows (del Gaudio et al., 2014):

$$X = Y + C \tag{2.1}$$

where: X is the *definiendum* (i.e. the term to be defined);$=$ is the *definitor* (i.e. a connective verb such as 'to be', 'consist');Y is the *definiens* (i.e. the genus phrase or, better described, the nearest superconcept);C are the *differentiae specificae*, the distinguishing characteristics that specify the distinction between one *definiendum* and another. According to equation 2.1, the sentence "Gnocchi are a varied family of dumpling in Italian cuisine" is composed as in Figure 2.5.

Definitional context identification is the task of automatically identifying the definitional context of a specific term within a document.

The task of definition identification is not limited to glossaries and encyclopaediae, but extended to other fields such as ontology learning (Gangemi et al., 2003), question answering (Saggion, 2004; Cui et al., 2007) and eLearning (Westerhout and Monachesi, 2007).

The majority of the work done on the topic relies on lexico-syntactic patterns (Saggion, 2004; Cui et al., 2007; Fahmi and Bouma, 2006; Degórski et al., 2008) that require manual annotation and/or manually written rules. A different approach has been taken with the use of Word Lattices. A Word Lattice (WL) is a directed acyclic graph (DAG) that is a representation of a segment, with a starting point that branches out to the other weighted tokens in the segment. WLs have been used for text retrieval (Carpineto and Romano, 2005) and machine translation (Dyer et al., 2008; Schroeder et al., 2009). Navigli and Velardi (2010) introduced Word-Class Lattices,

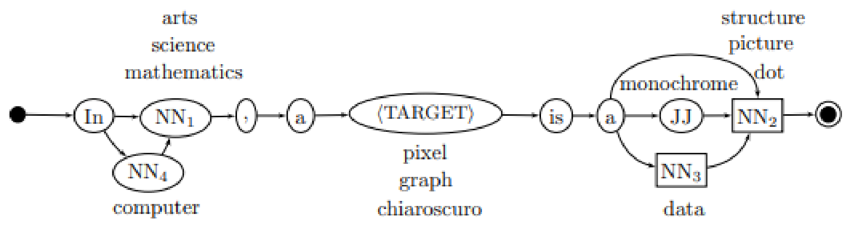


Figure 2.6: Example of a Word-Class Lattice that clusters three sentences (Navigli and Velardi, 2010).

a generalization of Word Lattices used to model textual definitions. The pipeline to construct WCLs from a text document consists of three steps:

1. Star patterns: each sentence is preprocessed by substituting all the tokens above a certain threshold with their corresponding part-of-speech (POS) tag, so that only the common parts of speech are kept as they are (e.g. prepositions and verbs), whereas the less common parts of speech (e.g. nouns and adjectives) are replaced with their respective POS tag, except for the *definiendum*, which is substituted by a TARGET label;

2. Sentence clustering: the preprocessed sentences that share similar structure are clustered together.

3. Word-Class Lattice Construction: the clusters are fed to a greedy alignment algorithm that constructs the Word-Class Lattices. Figure 2.6 displays an example of a WCL for three sentences: "In arts, a chiaroscuro is a monochrome picture.", "In mathematics, a graph is a data structure that consists of...", "In computer science, a pixel is a dot that is part of a computer image.".

In their approach, two models were evaluated: WCL-1, which learns all the fields at once (i.e. *definiendum* field, *definitor* field, *definiens* field and *differentiae specificae* field); WCL-3, which learns the fields separately. The latter model has shown the best results on accuracy.

The dataset with which they performed an experimental evaluation on WCLs is relevant as we will use it to train our models. This process is described in Section 3.2.

2.4 Automatic Definitions Linking

Wikification is the task of identifying the keywords in an input document and linking them to their entity page. The core of the task is to develop an efficient entity annotator that is able to select the keywords that will become the anchors in a text and link them to their corresponding entity pages.

Among the body of literature on the topic (Kulkarni et al., 2009; del Gaudio et al., 2014; Ratinov et al., 2011; Cucerzan, 2007), four systems have stood out.

The first known work is Wikify! (Mihalcea and Csomai, 2007), a system based on two stages:

1. Detection: keywords are identified using link probability, defined as the number of Wikipedia articles that use a specific term as an anchor divided by the number of articles that mention it;

2. Disambiguation: the context surrounding the term is extracted and compared to the training examples from Wikipedia.

Following up, Milne and Witten (2008) improved the approach by using disambiguation to inform detection:

1. Context pages identification: given an input text, find the pages linked by non-ambiguous spots, i.e. spots that link to only one page;

2. Relatedness: measure the relatedness between two pages according to the overlap between their in-linking Wikipedia pages;

3. Coherence: measure the coherence of one page with the other context pages.

Another system called TAGME (Ferragina and Scaiella, 2010) disambiguates mentions with a voting scheme that links it to its top-scoring candidate entity page:

1. Preprocessing: extracting anchors from Wikipedia pages and gathering all senses;

2. Anchor disambiguation: by computing a score for each possible sense of an anchor;

3. Anchor pruning: discarding anchors that are not considered candidates after the disambiguation phase. This is done by taking into account the link probability of an anchor and the coherence of its candidate annotation with respect to the other anchors' candidate annotations.

TAGME has later evolved into WAT (Piccinno and Ferragina, 2014). The approach stays the same, but the three-steps pipeline has been modified to be more efficient:

1. Spotting: produce a set of possible mentions and a list of candidate entities for every mention;

2. Disambiguation: associate a score to each candidate entity page (based on TAGME voting scheme);

3. Pruning: discarding unsuitable anchors (based on TAGME but optimized).

The concepts that constitute the basis of the aforementioned models served as baseline for the second aim of this research, discussed in Chapter 4.

Chapter 3

Definitional Contexts Identification

The first aim of this research is to train two supervised models so that they are able to identify the definitional contexts in Wikipedia articles in the Italian and English language. To do so, the two models were trained on a freely available, manually annotated dataset and tested on two Wikipedia sets of articles extracted from two food-related ad-hoc corpora in Italian and English. Section 3.1 introduces the two corpora used for the manual evaluation of the trained models' performance. Section 3.2 presents the metrics used to evaluate the results. Section 3.3 introduces the proposed models to train. Section 3.4 presents the results after the training. Section 3.5 discusses the results obtained when the models are fed with the sets of articles.

3.1 The DIT-WF-22 Corpora

Given the unavailability of freely accessible food-related Wikipedia articles, we built two corpora by parsing the Wikipedia Italian and English dumps from July 2021 with the technique explained in section 2.2. DIT-WF-22-IT, a corpus of Italian food-related Wikipedia articles and DIT-WF-22-EN, a corpus of English food-related Wikipedia articles, are thus presented in this section.

Italian Categories		
antipasti	secondi piatti	contorni
primi piatti	piatti unici	dolci

Table 3.1: Branch of categories for food-related articles in the Italian Wikipedia.

3.1.1 Wikipedia as a Source of In-Domain Comparable Corpora

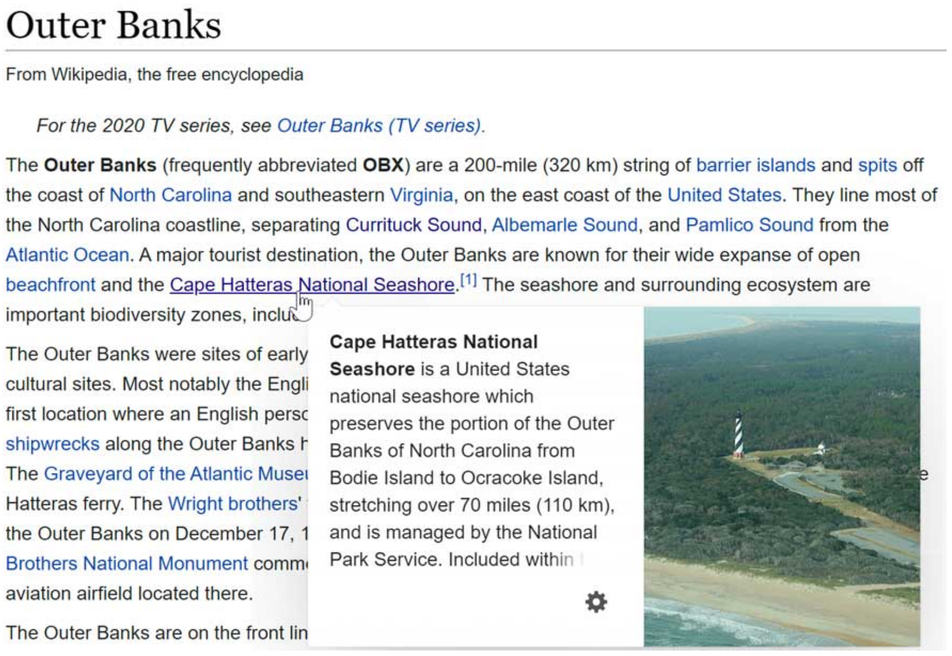
With contents currently available in 314 languages[1], Wikipedia is the largest encyclopedia in the World Wide Web. What makes it extremely useful are the wikilinks and the categories. Wikilinks are hyperlinks that connect one relevant term to its corresponding Wikipedia article. Figure 3.1 shows an example. Located at the bottom of articles, categories are wikilinks that direct the user to a category page listing articles or other pages that belong to a specific category[2] (or domain). They play a crucial role in the extraction of domain-specific articles. Each category belongs to a branch of the category tree[3], where general concepts belong to the supercategories (or higher levels) and the specific concepts belong to the subcategories (or lower levels). For instance, `Category:Pork dishes` is a subcategory that belongs to the supercategory of `Category:Meat dishes`.

To build the DIT-WF-22-IT corpus, we selected the categories of Table 3.1.

The reason for choosing the aforementioned categories is that most of the dishes in the Italian Wikipedia belong to subcategories under the supercategories listed in Table 3.1, which in turn are subcategories under the supercategory of `Category:Portate di cucina` (the assumption was confirmed by a manual search).

Being the categories of the food domain in the English version less organized, we carried out a broader selection in order to get as many food-related matches as possible. The chosen categories are displayed in Table 3.2.

[1]`https://meta.wikimedia.org/wiki/List_of_Wikipedias`; last visit 31.01.2022
[2]`https://en.wikipedia.org/wiki/Help:Categories`; last visit 14.01.2022
[3]`https://en.wikipedia.org/wiki/Special:CategoryTree`; last visit 14.01.2022


Outer Banks

From Wikipedia, the free encyclopedia

For the 2020 TV series, see Outer Banks (TV series).

The **Outer Banks** (frequently abbreviated **OBX**) are a 200-mile (320 km) string of barrier islands and spits off the coast of North Carolina and southeastern Virginia, on the east coast of the United States. They line most of the North Carolina coastline, separating Currituck Sound, Albemarle Sound, and Pamlico Sound from the Atlantic Ocean. A major tourist destination, the Outer Banks are known for their wide expanse of open beachfront and the Cape Hatteras National Seashore.[1] The seashore and surrounding ecosystem are important biodiversity zones, inclu

The Outer Banks were sites of early
cultural sites. Most notably the Engli
first location where an English pers
shipwrecks along the Outer Banks h
The Graveyard of the Atlantic Muse
Hatteras ferry. The Wright brothers'
the Outer Banks on December 17, 1
Brothers National Monument comm
aviation airfield located there.

The Outer Banks are on the front lin

Cape Hatteras National Seashore is a United States national seashore which preserves the portion of the Outer Banks of North Carolina from Bodie Island to Ocracoke Island, stretching over 70 miles (110 km), and is managed by the National Park Service. Included within

Figure 3.1: Example of a wikilink in the Wikipedia article on outer banks pointing to the article of Cape Hatteras National Seashore.

3.1.2 In-Domain Articles Preprocessing

We parsed the Italian and English Wikipedia dumps from the 20th of July 2021 (the latest versions at the time of downloading)[4].

Both for the Italian and English dumps, we extracted each article containing at least one of the categories in Table 3.1 or Table 3.2 for the Italian and English language respectively[5]. For each Wikipedia article we extracted the necessary body of text enclosed in XML tags and stored it into a txt file.

[4]A Wikipedia dump (`https://dumps.wikimedia.org/`; last visit 22.01.2022) is a copy of all Wikipedia pages existing in the form of Wikitext. Being the number of Wikipedia pages ever-growing, the dumps are updated on a regular basis. The Wikitext format is in XML and presents a rigid structure, which makes it easier to extract the data that is needed (Nothman, 2008).

[5]The implementation is derived from the freely available Github repository of Will Koehrsen: `https://github.com/WillKoehrsen/wikipedia-data-science/blob/master/notebooks/Downloading%20and%20Parsing%20Wikipedia%20Articles.ipynb`; last visit 10.10.2021

English Categories		
Italian cuisine	Cuisine of Calabria	Cuisine of Abruzzo
Cuisine of Apulia	Cuisine of Basilicata	Cuisine of Calabria
Cuisine of Campania	Cuisine of Emilia-Romagna	Cuisine of Lazio
Cuisine of Liguria	Cuisine of Lombardy	Cuisine of Marche
Cuisine of Molise	Cuisine of Piedmond	Cuisine of Sardinia
Cuisine of Sicily	Cuisine of South Tyrol	Cuisine of Tuscany
Cuisine of Umbria	Cuisine of Veneto	Cuisine of Aosta Valley
Egg dishes	Flower dishes	Fruit dishes
Ginger dishes	Grain dishes	Meat dishes
Mushroom dishes	Noodle dishes	Nut dishes
Pasta dishes	Tofu dishes	Tuber dishes
Vegetable dishes	Spaghetti dishes	Neapolitan cuisine
Potato dishes	Fish dishes	Italian desserts
Appetizers	Desserts	

Table 3.2: Set of categories for food-related articles in the English Wikipedia.

	DIT-WF-22-IT	**DIT-WF-22-EN**
Number of Articles	2,054	1,923
Number of Tokens	780,996	1,170,360
Average Number of Tokens	380	608

Table 3.3: Statistics of the DIT-WF-22-IT and DIT-WF-22-EN corpora.

For more details, see Appendix A.

The result are two corpora containing only food-related articles. Table 3.3 displays the information on both corpora.

3.2 A Framework for Definitional Context Identification

For the purpose of this research we chose the manually annotated dataset WIKI-DC-EN-2010[6] (Navigli et al., 2010) used to perform an experimental evaluation on the WCL algorithm (Navigli and Velardi, 2010). The corpus was built by randomly extracting the first sentences of Wikipedia articles

[6]`http://lcl.uniroma1.it/wcl/`; last visit 15.10.2021

Positive Segments	**Negative Segments**	**Total Segments**
1,872	2,847	4,719

Table 3.4: Class distribution of WIKI-DC-EN-2010 (Navigli et al., 2010).

from different categories. The segments were labelled as “positive” or “negative”, where the first label means that the segment is a definitional context, the second label means that the segment is not a definitional context. Table 3.4 shows the class distribution. For the purpose of this task, the labels were changed to the binary annotation of 0 (not definitional context) and 1 (definitional context).

The metrics used to assess the performance are F$_1$-measure and accuracy. Since to calculate the F$_1$-measure precision and recall are required, they will be introduced first.

Both precision and recall are defined in terms of the confusion matrix in Figure 3.2. True negatives are all the negative instances that were correctly predicted as negative; false negatives are all the positive instances that were incorrectly predicted as negative; false positives are all the negative instances that were incorrectly predicted as positive; true positives are all the positive instances that were correctly predicted as positive.

Precision is defined as:

$$Precision = \frac{True\ Positives}{True\ Positives + False\ Positives} \tag{3.1}$$

		Predicted	
		Negative	**Positive**
Actual	**Negative**	True Negatives (TN)	False Positives (FP)
	Positive	False Negatives (FN)	True Positives (TP)

Figure 3.2: Confusion matrix relating actual and predicted labels in a binary classification setting.

Therefore, precision determines how many positive instances are actually positive out of all predicted positives.

Recall is defined as:

$$Recall = \frac{TP}{TP + FN} \tag{3.2}$$

Therefore, recall calculates how many actual positive instances the model has predicted out of all positive cases.

The F_1-measure is the harmonic mean of precision and recall:

$$F1 = 2\frac{Precision \cdot Recall}{Precision + Recall} \tag{3.3}$$

Accuracy is defined as:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.4}$$

Therefore, accuracy evaluates how many instances are correctly predicted out of all the predictions.

3.3 BERT models

We decided to train two BERT models (Devlin et al., 2018), one for English and one for Italian. For the first language we chose `bert-base-cased`, a model pretrained on English language. For the second language we chose `bert-base-multilingual-cased`, a model pretrained on the top 104 languages with the most articles in the Wikipedia. They are both encoder models (their functioning is explained in Section 2.1.1).

When fine-tuned for this task, the text the models are fed with must be split sentence-wise in order for the models to classify it. They assign each sentence a negative and a positive score, both ranging from 0 to 1: if the first score is higher, the sentence can be deemed negative; if the second score is higher, the sentence can be deemed positive.

3.4 Experiments and Results

The manually annotated dataset WIKI-DC-EN-2010 (Navigli et al., 2010) was split into 80% for the training, 10% for the testing and 10% for the validation and the models `bert-base-cased` and `bert-base-multilingual-cased` (Devlin et al., 2018) were trained on it.

	F_1-measure	**Accuracy**
bert-base-cased	96.08	96.82
bert-base-multilingual-cased	97.66	98.09

Table 3.5: F_1-measure and accuracy results for the task of definitional context identification of the two BERT models.

Model	**Mean F_1-measure**
bert-base-cased	97.63 ± 0.95
bert-base-multilingual-cased	97.34 ± 1.02

Table 3.6: Mean F_1-measure for the 10-fold cross validation performed on the manually annotated dataset (Navigli et al., 2010).

Table 3.5 shows the performance of our models on the validation set. To give perspective to the results, the developers of the corpus report a performance of 75.23 on the F_1-measure and 83.84 on accuracy[7].

To further confirm the performance of our model, cross validation was applied. Cross validation is a technique that assesses how the results of a statistical analysis generalize to an independent dataset[8]. There are different types of cross validation techniques and the one used for this research is k-fold cross validation. For this process, the dataset is split into k equal-sized folds (also called partitions), one fold is used for testing and the other *k-1* folds are used for training. Therefore, in 10-fold cross validation, the dataset is split into 10 folds, where one is used for testing and the other nine for training. Table 3.6 displays the results from the cross validation for our models. F_1-measure mean is the mean of the ten F_1-measures from the single partitions; standard deviation is the square root of the variance: the lower the value, the closer the F_1-measures of each partition are to each other, which indicates that the models are stable.

[7]Unfortunately, training and testing partitions of WCL-3 have not been published, so our results are not directly comparable.

[8]`https://en.wikipedia.org/wiki/Cross-validation_(statistics)#k-fold_cross-validation`; last visit 15.01.2022

3.5 The Definition Extractor at Work

To analyse how well our definition extractor performs when fed with real documents, we tested the models on two subsets of the DIT-WF-22-IT and DIT-WF-22-EN corpora and we manually evaluated their classifications of definitional contexts. To achieve the goal of extracting the right definitional contexts, we took into account two output scenarios:

- Top score: the sentence with the highest BERT score overall in the article;
- Pos: it is based on the position of the sentences. The first sentence in the article whose BERT score is higher than 0.6[9] is picked as the candidate.

The subsets with which we tested the two models consisted of 30 randomly picked articles from each corpus[10].

In order to obtain the desired output to analyse, an article is fed to the model sentence-wise using the `spaCy` library (Honnibal and Montani, 2017)[11] and the sentences are ranked according to their score.

We evaluated the models' performance on real documents with the *precision at k* metric. P@k, with k=1, calculates how many instances that the model has labeled as definitions by assigning them a high positive score are actually definitions. Table 3.7 displays the results. Both models' performances are significantly higher with the "pos" approach compared to the "top score" method.

Tables 3.8, 3.9, 3.10 and 3.11 display four examples of the scoring applied by the models: the first two have been extracted for English, the latter two for Italian. The first sentence follows the "top score" approach, the last sentence follows the "pos" approach.

[9]A first manual analysis of a sample of random articles showed how most of the positive scores were above 0.7. To ensure that the classification algorithm included all the potential positive sentences (i.e. potential definitional contexts), the threshold of 0.6 suited best our goal.

[10]The spreadsheets with the full subsets can be viewed at `https://docs.google.com/spreadsheets/d/125T-KKFJrI_WeQu3oSHMEzV28XYXamJw4g88v38dv0Y/edit#gid=0` for English and `https://docs.google.com/spreadsheets/d/1hB4G1hjdvuVtxgHSOvmYj9dKOvqSRJ9vT4YtbrcwSG8/edit#gid=0` for Italian.

[11]`http://spacy.io/`; last visit 01.02.2022

Model	**Top score**	**Pos**
`bert-base-cased`	0.76	0.96
`bert-base-multilingual-cased`	0.86	0.90

Table 3.7: P@1 results comparison between "top score" and "pos" approach on both models.

Score	**Approach**	**Sentence**
0.999	Top score	Picada is a type of tapas eaten in Argentina and Uruguay, usually involving only cold dishes, such as olives, ham, salami, mortadella, bologna,different types of cheese, marinated eggplants and red pimentos, sardines, nuts, corn puffs, fried wheat flour sticks, potato chips, and sliced baguette.
0.999	Pos	A tapa () is an appetizer or snack in Spanish cuisine.

Table 3.8: Example where the "top score" and "pos" scenarios match in the article of tapas.

In Table 3.8 the sentence with the top BERT score matches the first positive sentence, extracted from the English article about tapas. In Table 3.9 the sentence with the top BERT score mismatches the first positive sentence, extracted from the English article of meringue. In Table 3.10 the sentence with the top BERT score matches the first positive sentence, extracted from the Italian article of *canederli* (semmelknödel). In Table 3.11 the sentence with the top BERT score mismatches the first positive sentence, extracted from the Italian article of *bastoncini di pesce* (fish fingers).

These examples confirm that the models work clearly best when constrained to the first sentence in the article surpassing the threshold. Although the "top score" approach showed a small difference compared to "pos" in the Italian model, the latter nonetheless performed generally better. It is to be noted that, by keeping in mind the concept of a definitional context presented in Section 2.3, all the extracted sentences can be considered definitions. Therefore, the models are technically correct and it's the Wikipedia articles that are misleading by containing multiple definitions, only one of which is the correct one for the *definiendum*. Since the aim is to identify the one corresponding definition of a term, extracting the first sentence in the article whose score is higher than 0.6 has proven to work better compared

Score	**Approach**	**Sentence**
0.999	Top score	Sucrose is a disaccharide made up of glucose and fructose.
0.999	Pos	Meringue (, ;) is a type of dessert or candy, often associated with Swiss, French, Polish and Italian cuisines, traditionally made from whipped egg whites and sugar, and occasionally an acidic ingredient such as lemon, vinegar, or cream of tartar.

Table 3.9: Example where the “top score” and “pos” scenarios mismatch in the article of meringue.

Score	**Approach**	**Sentence**
0.999	Top score	I canéderli (in tedesco Semmelknödel) sono degli Knödel (grossi gnocchi) composti di un impasto a composizione variabile di pane raffermo.
0.999	Pos	I canéderli (in tedesco Semmelknödel) sono degli Knödel (grossi gnocchi) composti di un impasto a composizione variabile di pane raffermo.

Table 3.10: Example where the “top score” and “pos” scenarios match in the article of *canederli*.

to the “top score”. “Pos” has thus been chosen for the final pipeline for the definition linking, explained in Chapter 4[12].

[12] Since we assumed that the parentheses in the sentences would lower the scoring of the models, we tried removing them, though we noticed no significant change in score. Thus, we decided to keep the parentheses.

Score	**Approach**	**Sentence**
0.998	Top score	Le “dita di pesce” (fish fingers) furono una ricetta di inizio Novecento pubblicata su una popolare rivista britannica [...]
0.993	Pos	I bastoncini di pesce sono un prodotto alimentare trasformato e commerciale, preparato con pesce bianco [...]

Table 3.11: Example where the “top score” and “pos” scenarios mismatch in the article of *bastoncini di pesce*.

Chapter 4

Automatic Definition Linking

The second aim of this research work is developed around the concept of *wikification*. Going beyond the basic process of linking a term to its corresponding Wikipedia article, we aim to link meaningful terms to their definitions in the corresponding Wikipedia article. A description of the theoretical and practical aspects of the wikification process is available in Section 2.4. Section 4.1 describes the two ad-hoc corpora used as input for the pipeline we propose in Section 4.2. The systems described in Section 2.4 served as basis for our simplified one. The systems' basic concepts were kept and adapted to our needs through a vanilla rule-based approach.

4.1 The DIT-GZ-22 Corpora

Given the lack of corpora of Italian and English recipes, two ad-hoc corpora were built by using a cooking website and preprocessing it for the purpose of this research. The creation of DIT-GZ-22-IT, a corpus of Italian recipes and DIT-GZ-22-EN, a corpus of English recipes, is thus presented in this section.

4.1.1 Recipes Retrieval

Following the steps described in section 2.2, we scraped Giallo Zafferano, a popular cooking website. The Italian version of the website has 4 main categories: *antipasti* (appetizers), *primi* (first courses), *secondi* (second courses) and *dolci* (desserts). Table 4.1 displays the four main categories. Each course

Antipasti	**Primi**	**Secondi**	**Dolci**
Antipasti veloci	Pasta fresca	Carne	Biscotti
Pizze e focacce	Pasta	Facili e veloci	Piccola pasticceria
Pane	Primi veloci	Pesce	Dolci veloci
Finger food	Grandi classici	Secondi vegetariani	Torte
Torte salate	Paste sfiziose	Contorni	Cioccolato
Ricette al forno	Gnocchi	Piatti unici	Torte veloci
Insalate	Riso e cereali	Fritti	Marmellate

Table 4.1: Main categories and their dishes in the Italian version.

	DIT-GZ-22-IT	**DIT-GZ-22-EN**
Number of Recipes	10,262	259
Average Number of Tokens	477±163	572±199
Vocabulary size	28,715	6,406

Table 4.2: Statistics of the DIT-GZ-22-IT and DIT-GZ-22-EN corpora.

is further divided into seven types of dishes, for a total of 36 categories. Under these categories are grouped the recipes.

The English version of the website presents a different structure in which the recipes are grouped under 6 main categories: latest recipes, appetizers, first courses, main courses, desserts, leavened products. There is no further division.

For each category, we downloaded the HTML version of all the recipes assigned to it.

4.1.2 Recipes Preprocessing

Once downloaded, we preprocessed each recipe. We carried out the preprocessing using the `BeautifulSoup` library. From the HTML text we extracted four parts enclosed in specific HTML tags and cleaned each of them. We then wrote the four parts into four txt files: titles.txt, presentation.txt, ingredients.txt, preparation.txt. We stored the files into a folder, so that each recipe has a unique folder. Appendix A includes further details.

Table 4.2 shows the statistics of the DIT-GZ-22-IT and DIT-GZ-22-EN corpora.

We applied the steps to build both corpora. We used the two corpora for the experiments carried out for the automatic definitions linking and further delineated in section 4.2.

4.2 Definitions Linking

For the purpose of this research the task of *wikification* has been modified according to our purpose. Limiting the range of domains to the food one has made the disambiguation and pruning phases more straightforward, with no need to score candidate entity pages nor prune anchors (as done in the pipelines of the systems explained in Section 2.4). These two phases could thus be joined into a single one.

We experimented with the titles of the recipes from the DIT-GZ-22-EN and DIT-GZ-22-IT corpora since they provide very short, unambiguous sentences, suitable for our work.

- Let t be the input text;
- Let G be the list of all possible n-grams;
- Let g be one n-gram $g \in G$;
- Let e be the entity page;
- Let H be the set of titles from the DIT-WF-22-EN and the DIT-WF-22-EN corpora;
- Let h be one title $h \in H$.

The pipeline in Figure 4.1 shows our approach to the automatic definitions linking task. Each box is a step of the overall approach, where the oval is the start/end, the parallelogram represents the input, the rectangle the process, the diamond indicates a yes/no decision to be made and the lines function as connectors between the stages. It is to be noted that the two stages highlighted in red, namely Language Detector and Definitions Linker, have not been covered yet.

1. **Language detection**: given an input text t, detect the language;
2. **N-gram creator**: produce G and sort each g from shortest to longest;

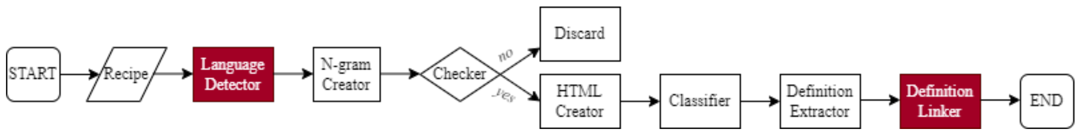


Figure 4.1: Flowchart of the full pipeline.

3. **Checker**: for every $g \in G$, check if a corresponding e exists inside the Wikipedia[1];
4. **HTML Extractor**: produce a minimalistic HTML version of t, pairing each g with its corresponding e enclosed in a HTML tag;
5. **Classifier**: if g is a $h \in H$, assign g the attribute `class=food`, otherwise assign it the attribute `class=other` to the HTML tag.
6. **Definition Extractor**: for each $h \in H$, a definitional context is extracted and assigned to its corresponding g;
7. **Definitions Linker**: for each g that has a corresponding $h \in H$, link the definitional context of the Wikipedia article.

The work described in the previous Chapters of this dissertation covers most of the steps of the aforementioned pipeline. This could serve as starting point for future tasks, explained in Section 5.2.

[1]Wikipedia API `https://github.com/martin-majlis/Wikipedia-API`; last visit: 27.01.2022

Chapter 5

Conclusions

After presenting the results obtained from the BERT models for the definitional contexts extraction and the pipeline for the definitions linking process, this chapter draws the conclusions on the overall process. Section 5.1 summarizes what has been done. Section 5.2 discusses promising departing points for follow up research.

5.1 Final Remarks

The two main objectives of this research are definitional contexts identification and automatic definitions linking in the food domain for the Italian and English language. To set up the basis of the work, we built four ad-hoc corpora, two from the Italian and English versions of a cooking website and two from the Italian and English dumps of Wikipedia. From the website we collected recipes and from the Wikipedia dumps we extracted only articles belonging to the food domain.

In order to accomplish the first goal, we trained two BERT models, one for each language, with a manually annotated corpus of Wikipedia sentences. We obtained a F_1-measure of 96.08 and 97.66 and an accuracy of 96.82 and 98.09. To test the models' performance on real documents, we fed each model with 30 randomly picked articles, one with Italian Wikipedia articles and one with English Wikipedia articles. Since one sentence is enough to be considered a definitional context for a term, we considered two output scenarios: "top score" and "pos". The first scenario considers only the sentence with the highest BERT score overall, the second scenario considers only the first sentence in the article whose BERT score is higher than 0.6. A manual evaluation showed that both models identified segments that can technically be considered definitional contexts according to Aristotle's equation, in both scenarios. Since the goal was to find the corresponding definition of a term inside a Wikipedia article about that term, the "pos" approach showed to work best compared to "top score" approach. These findings were confirmed by P@1 scores of 0.96 for the model fed with English articles and 0.90 for the model fed with Italian articles, which show that the models are stable.

The second goal of this work was partly fulfilled by implementing a pipeline. When the algorithm is fed with the title of a recipe as input text, it detects the language (Italian or English in this case), creates a list of n-grams, checks if to each n-gram a Wikipedia article exists and outputs a minimalistic HTML version of the input text with the n-grams linked to their corresponding definitional contexts in the existing Wikipedia articles. The pipeline we have developed so far covers most of the stages as we have thoroughly described them in the previous Chapters.

5.2 Future Work

One of the future tasks that can be addressed is completing the pipeline with the two missing stages, namely Language Detection and Definitions Linking. Language Detection is the very first step of the pipeline after the algorithm is fed with the input text. Definitions Linking goes a step beyond the *wikification* process and aims at linking a term to its corresponding definitional context in a Wikipedia article.

Another interesting work that could be carried out is cross-language definitions linking. When given an input text in language A, the system could cross-reference and extract the definitions in language B.

The last task we put forward in this dissertation is of an application for the catering industry: a cross-language application that gives definitions of dishes from language A to language B could be helpful for non-native speakers and make reading a menu quicker and more accessible. This research work shows that it is possible to design and develop such an application.

Bibliography

Claudio Carpineto and Giovanni Romano. Using concept lattices for text retrieval and mining. In *Formal Concept Analysis*, Berlin, Heidelberg, 2005. Springer. doi: https://doi.org/10.1007/11528784_9.

Silviu Cucerzan. Large-scale named entity disambiguation based on Wikipedia data. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 708–716, Prague, Czech Republic, June 2007. Association for Computational Linguistics. URL `https://aclanthology.org/D07-1074`.

Hang Cui, Min-Yen Kan, and Tat-Seng Chua. Soft pattern matching models for definitional question answering. *ACM Transactions on Information Systems*, 25(2):8–es, 2007. doi: 10.1145/1229179.1229182.

Łukasz Degórski, Michał Marcińczuk, and Adam Przepiórkowski. Definition extraction using a sequential combination of baseline grammars and machine learning classifiers. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*, Marrakech, Morocco, May 2008. European Language Resources Association (ELRA). URL `http://www.lrec-conf.org/proceedings/lrec2008/pdf/213_paper.pdf`.

Rosanna del Gaudio, Gustavo E. A. P. A. Batista, and António Horta Branco. Coping with highly imbalanced datasets: A case study with definition extraction in a multilingual setting. *Natural Language Engineering*, 20(3): 327–359, 2014. doi: 10.1017/S1351324912000381.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT:

pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018. URL `http://arxiv.org/abs/1810.04805`.

Christopher Dyer, Smaranda Muresan, and Philip Resnik. Generalizing word lattice translation. In *Proceedings of ACL-08: HLT*, pages 1012–1020, Columbus, Ohio, June 2008. Association for Computational Linguistics. URL `https://aclanthology.org/P08-1115`.

Ismail Fahmi and Gosse Bouma. Learning to identify definitions using syntactic features. In *Proceedings of the Workshop on Learning Structured Information in Natural Language Applications*, 2006. URL `https://aclanthology.org/W06-2609`.

Paolo Ferragina and Ugo Scaiella. Tagme: On-the-fly annotation of short text fragments (by wikipedia entities). CIKM '10, page 1625–1628, New York, NY, USA, 2010. Association for Computing Machinery. ISBN 9781450300995. doi: 10.1145/1871437.1871689. URL `https://doi.org/10.1145/1871437.1871689`.

Aldo Gangemi, Roberto Navigli, and Paola Velardi. The ontowordnet project: Extension and axiomatization of conceptual relations in wordnet. In *Proceedings of the International Conference on Ontologies, Databases and Applications of Semantics (ODBASE 2003)*, pages 820–838, Catania, Italy, 2003.

Kai Hakala and Sampo Pyysalo. Biomedical named entity recognition with multilingual BERT. In *Proceedings of The 5th Workshop on BioNLP Open Shared Tasks*, pages 56–61, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-5709. URL `https://aclanthology.org/D19-5709`.

Lane Hobson, Cole Howard, and Hannes Max Hapke. *Natural Language Processing in Action*. Manning Publications, 2019.

Matthew Honnibal and Ines Montani. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. 2017.

Moaiad Ahmad Khder. Web scraping or web crawling: State of art, techniques, approaches and application. *International Journal of Advances in Soft Computing I& Its Applications*, 13(3):144–168, 2021.

Sayali Kulkarni, Amit Singh, Ganesh Ramakrishnan, and Soumen Chakrabarti. Collective annotation of wikipedia entities in web text. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '09, page 457–466, New York, NY, USA, 2009. Association for Computing Machinery. ISBN 9781605584959. doi: 10.1145/1557019.1557073. URL `https://doi.org/10.1145/1557019.1557073`.

Harish Tayyar Madabushi, Elena Kochkina, and Michael Castelle. Cost-sensitive BERT for generalisable sentence classification with imbalanced data. *CoRR*, abs/2003.11563, 2020. URL `https://arxiv.org/abs/2003.11563`.

Rada Mihalcea and Andras Csomai. Wikify! linking documents to encyclopedic knowledge. In *Proceedings of the Sixteenth ACM Conference on Conference on Information and Knowledge Management*, CIKM '07, page 233–242, New York, NY, USA, 2007. Association for Computing Machinery. ISBN 9781595938039. doi: 10.1145/1321440.1321475. URL `https://doi.org/10.1145/1321440.1321475`.

David Milne and Ian H. Witten. Learning to link with wikipedia. In *Proceedings of the 17th ACM Conference on Information and Knowledge Management*, CIKM '08, page 509–518, New York, NY, USA, 2008. Association for Computing Machinery. ISBN 9781595939913. doi: 10.1145/1458082.1458150. URL `https://doi.org/10.1145/1458082.1458150`.

Roberto Navigli and Paola Velardi. Learning word-class lattices for definition and hypernym extraction. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1318–1327, Uppsala, Sweden, July 2010. Association for Computational Linguistics. URL `https://aclanthology.org/P10-1134`.

Roberto Navigli, Paola Velardi, and Juana Maria Ruiz-Martínez. An annotated dataset for extracting definitions and hypernyms from the web. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*, Valletta, Malta, May 2010. European Language Resources Association (ELRA). URL `http://www.lrec-conf.org/proceedings/lrec2010/pdf/20_Paper.pdf`.

Joel Nothman. Learning named entity recognition from wikipedia. 2008.

Emil Persson. *Evaluating tools and techniques for web scraping*. PhD thesis, 2019. URL `http://urn.kb.se/resolve?urn=urn:nbn:se:kth:diva-271206`.

Francesco Piccinno and Paolo Ferragina. From tagme to wat: A new entity annotator. In *Proceedings of the First International Workshop on Entity Recognition & Disambiguation*, ERD '14, page 55–62, New York, NY, USA, 2014. Association for Computing Machinery. ISBN 9781450330237. doi: 10.1145/2633211.2634350. URL `https://doi.org/10.1145/2633211.2634350`.

Chen Qu, Liu Yang, Minghui Qiu, W. Bruce Croft, Yongfeng Zhang, and Mohit Iyyer. Bert with history answer embedding for conversational question answering. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR'19, page 1133–1136, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450361729. doi: 10.1145/3331184.3331341. URL `https://doi.org/10.1145/3331184.3331341`.

Lev Ratinov, Dan Roth, Doug Downey, and Mike Anderson. Local and global algorithms for disambiguation to Wikipedia. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 1375–1384, Portland, Oregon, USA, June 2011. Association for Computational Linguistics. URL `https://aclanthology.org/P11-1138`.

Leonard Richardson. Beautiful soup documentation. *April*, 2007.

Horacio Saggion. Identifying denitions in text collections for question answering. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC 2004)*, Lisbon, Portugal, 2004.

Josh Schroeder, Trevor Cohn, and Philipp Koehn. Word lattices for multi-source translation. In *EACL*, 2009.

Fábio Souza, Rodrigo Frassetto Nogueira, and Roberto de Alencar Lotufo. Portuguese named entity recognition using BERT-CRF. *CoRR*, abs/1909.10649, 2019. URL `http://arxiv.org/abs/1909.10649`.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008, 2017.

Eline Westerhout and Paola Monachesi. Extraction of dutch definitory contexts for elearning purposes. *Lot Occasional Series*, 7:219–234, 2007.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online, October 2020. Association for Computational Linguistics. URL `https://www.aclweb.org/anthology/2020.emnlp-demos.6`.

Bo Zhao. *Web Scraping*, pages 1–3. 05 2017. ISBN 978-3-319-32001-4. doi: 10.1007/978-3-319-32001-4_483-1.

Appendix A

Recipes and Articles Storage

For the purpose of this research work it was necessary to create four ad-hoc corpora. The DIT-GZ-22-IT and DIT-GZ-22-EN corpora consist of Italian and English recipes respectively. The DIT-WF-22-IT and DIT-WF-22-EN corpora consist of Italian and English Wikipedia articles that belong to the food domain. For each corpus, the text were downloaded and preprocessed according to our needs.

For the first two corpora a popular cooking website called Giallo Zafferano was scraped. The HTML of the recipes has a specific structure that allowed a quite straightforward preprocessing. Table A.1 exemplifies the preprocessing explained in section 2. The four parts of each recipe were cleaned from nested HTML tags and then stored into TXT files. Table A.1 displays an example of the 4 cleaned files contained in a folder of the recipe for *panna cotta*: **title.txt** contains the title of the recipe; **presentation.txt** contains the opening paragraph that introduces the recipe to the reader; **ingredients.tsv** contains the ingredients used for the recipe and each ingredient is tab-separated from its quantity; **preparation.txt** contains the steps to make the recipe. The four files were stored into a folder, so that each recipe has one folder with the TXT files that contain all the necessary information. The folders were named `gz_language_number`, where `gz` stands for Giallo Zafferano (the website from which recipes were downloaded), `language` is the language of the recipes, either 'it' for Italian or 'en' for English, `number` stands for a sequential number.

For the latter two corpora, each Wikipedia article that belongs to the food domain was cleaned by removing the "See also" section up until the bottom of the article (because considered not essential), some other residual noise and stored into a TXT file, the title separated from the text with a newline.

gz_it_0083
title.txt
Spaghetti alle vongole
presentation.txt
Direttamente dalla tradizione campana gli spaghetti alle vongole, decisamente uno dei più importanti piatti della cucina italiana e più amati tra i primi piatti di pesce. Una ricetta semplicissima dal meraviglioso sapore di mare, che nasconde qualche piccolo segreto per una perfetta riuscita. Vongole polpose, spolverata di prezzemolo e la deliziosa cremina che si crea naturalmente con l'amido della pasta, fanno degli spaghetti alle vongole una vera prelibatezza. E' un piatto perfetto per ogni occasione, dalla cena tra amici al piatto della domenica fino alla portata perfetta per il cenone di Natale della Vigilia o per Capodanno.
ingredients.tsv
ingredient quantity Spaghetti 320 g Vongole 1 kg Aglio 1 spicchio Prezzemolo 1 mazzetto Olio extravergine d'oliva q.b. Pepe nero q.b. Sale fino q.b. Sale grosso per le vongole q.b.
preparation.txt
Per preparare gli spaghetti alle vongole, cominciate dalla pulizia. Assicuratevi che non ci siano gusci rotti o vuoti, andranno scartati. Passate poi a batterle contro il lavandino, o eventualmente su un tagliere [1]. Questa operazione è importante per verificare che non ci sia sabbia all'interno: le bivalve sane resteranno chiuse, quelle piene di sabbia invece si apriranno [1]. Poi ponete le vongole in un colapasta poggiato su una ciotola e sciacquatele [2]. Ponete il colapasta in una ciotola e aggiungete abbondante sale grosso. Lasciate in ammollo le vongole per 2-3 ore [3]. Trascorso il tempo le vongole avranno spurgato eventuali residui di sabbia. In un tegame mettete a scaldare un po' d'olio [4]. Poi aggiungete uno spicchio d'aglio e, metre questo si insaporisce, scolate bene le vongole, sciacquatele e tuffatele nel tegame caldo [5]. Chudete con il coperchio e lasciate cuocere per qualche minuto a fiamma alta [6]. [...]

Table A.1: Example of TXT files contained in a folder of the recipe for *spaghetti alle vongole*.

Glossary

anchor Also called spots or mentions, they are a sequence of one or more keywords that are displayed in the document and serve as anchor for the wikilink that directs to its corresponding Wikipedia page, or entity page. 25, 26

entity Unambiguous identifier, or term. 25

entity page Wikipedia article that contains information on one single entity. 25, 41, 55

HTML HyperText Markup Language. 21, 22

HTTP HyperText Transfer Protocol. 21

n-gram a contiguous sequence of n words. A 1-gram is a single word, a 2-gram is a sequence of two words clustered together, and so on. 41

nested HTML tag a HTML tag that is inside another HTML tag.. 51

P@k precision at k. 34

sense Wikipedia article. 25

token a sequence of characters grouped together to form a semantic unit. 19

Wikipedia dump large collection of Wikipedia articles in one language. 22

XML Extended Markup Language. 22, 29