FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Assignment of bachelor's thesis

Title:	Product Compatibility Detection from Product Description
Student:	Tomáš Bánhegyi
Supervisor:	Ing. Miroslav Čepek, Ph.D.
Study program:	Informatics
Branch / specialization:	Knowledge Engineering
Department:	Department of Applied Mathematics
Validity:	until the end of summer semester 2022/2023

Instructions

One of the important tasks in ECommerce is to identify compatible products - for example, to recommend spare parts or consumables compatible with the main product or to filter eshop items. Unfortunately, the compatibility information is not always available in a machine-readable format. On the other hand, compatibility information is typically available in free text as a product description. The aim of this thesis is to leverage Natural Language Processing techniques of Artificial Intelligence to identify the compatibility information from the free text description.

Review Natural Language Processing (NLP) techniques, specifically the Named Entity Recognition, the Relationship Extraction and the Question Answering. Identify an appropriate dataset containing a product description along with the product compatibility information (for example, Amazon printer cartridges and tonner description). Apply selected NLP techniques to identify the compatible products.

Electronically approved by Ing. Karel Klouda, Ph.D. on 1 February 2022 in Prague.

Bachelor's thesis

PRODUCT COMPATIBILITY DETECTION FROM PRODUCT DESCRIPTION

Tomáš Bánhegyi

Faculty of Information Technology
Department of Applied Mathematics
Supervisor: Ing. Miroslav Čepek, Ph.D.
May 12, 2022

Contents

List of Figures

List of Tables

I thank my supervisor, Ing. Miroslav Čepek, Ph.D., for his guidance, support, regular consultations and insightful comments. I thank my reviewer Ing. Luděk Kopáček, Ph.D. for his valuable feedback. Finally, I thank my family for their support and patience.

Declaration

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

I acknowledge that my thesis is subject to the rights and obligations stipulated by the Act No. 121/2000 Coll., the Copyright Act, as amended, in particular that the Czech Technical University in Prague has the right to conclude a license agreement on the utilization of this thesis as a school work under the provisions of Article 60 (1) of the Act

In Prague on May 12, 2022 .....

Abstrakt

Táto bakalárska práca sa zameriava na získavanie kompatibility produktov z produktových popisov. Riešenie využíva známe modely strojového učenia z oblasti spracovania prirodzeného jazyka. Špecificky sa práca zameriava na úlohy rozpoznávania pomenovaných entít, extrakcie vzťahov a odpovedania na otázky.

Vhodný dataset musí obsahovať anotácie pre pomenované entity a vzťahy medzi nimi. Po vytvorení datasetu sa aplikujú vybrané modely strojového učenia. Podarilo sa mi extrahovať informácie o kompatibilite produktov so skóre 62.30%, pričom som aplikoval len rozpoznávanie pomenovaných entít a extrakciu vzťahov. Rozhodli sme sa vynechať úlohu odpovedania na otázky, pretože by sme presiahli rozsah bakalárskej práce.

Táto práca prináša riešenie pre využitie predtrénovaných modelov, za účelom analýzy popisov produktov a získania informácie o ich kompatibilite. V závere sú popísané možnosti pre ďalší výskum. A ako príloha je uvedený podrobný popis konfiguračného súboru použitého pre náš model.

Klíčová slova spracovanie prirozeného jazyka, rozpoznávanie pomenovaných entít, extrakcia vzťahov, odpovedanie na otázky, analýza kompatibility, transformery

Abstract

This bachelor thesis focuses on processing product descriptions to extract product compatibility information. The solution uses known machine learning models from the natural language processing field and its subtasks named entity recognition, relationship extraction, and question answering.

A suitable dataset must contain annotations specifying the named entities and their relationships. After creating the dataset, there are applied selected machine learning models. I extracted product compatibility information with a 62.30% score, using just named entity recognition and relationship extraction. We decided to skip the question answering task because it would be out of scope for this bachelor thesis.

This thesis brings a solution for leveraging pre-trained models to analyse product descriptions and extract their compatibility. In summary, there are described the possibilities for further research. As an appendix, there is a detailed description of the configuration file used for our model.

Keywords natural language processing, named entity recognition, relationship extraction, questions answering, compatibility analysis, transfromers

List of abbreviations

NLP	Natural Language Proccessing
NER	Named Entity Recognition
NE	Named Entity
IE	Information Extraction
REL	Relationship Extraction
RNN	Recurrent Neural Network
LSTM	Long short-term memory

Chapter 1
Introduction

Online shopping has been on the rise in recent years. The increase in the number of companies selling their products online increased the variety of products sold online. This trend introduces various challenges. One of them is that regular customers shopping online need to study the products they want to buy. When shopping for spare parts, add-ons, or consumables, the issue becomes even more complex. The customer needs to find products compatible with the already owned products.

There usually is no option for the customer to consult their buying decisions in online sales. If there is an option to consult their decision, they can usually take advantage only of text-based communication. Moreover, the customer cannot be sure if he is talking to a real person or just a chatbot. In the case of a chatbot, the seller would need to have solved the problem of product compatibility. On the other hand, when shopping in-store, there is always a salesperson that understands the product portfolio and can advise the customer right on the spot.

This thesis tries to solve the problem by transferring the burden of research from the customer to the machine. The main task is extracting the compatibility of various products from provided text descriptions. If the seller wants to solve this task manually, it would require a significant effort to create a knowledge base of product compatibility. Moreover, after the creation, there would still need to be someone maintaining the system and updating compatibility for thousands of products. This manual solution might not be feasible for the seller.

We want to solve this problem by taking advantage of the existing product descriptions, combined with the existing machine learning models and solutions in the natural language processing field, thus creating a less human-dependent solution. Our solution is training a machine learning model capable of answering the question of compatibility whilst taking the product descriptions as input. The seller could benefit from this model by analyzing their product portfolio. The analysis will build the knowledge base of relationships between the products. With the knowledge base, the seller can present compatibility information more accessibly to the customer.

In the first part of this thesis, we will review natural language processing techniques and their contribution to product compatibility detection. Three tasks will be beneficial for this thesis, and those are: Named Entity Recognition, Relationship Extraction, and Questions Answering. These tasks are introduced to the reader in separate sections explaining their contribution to the main task.

The second part focuses on finding and preparing a suitable dataset. When looking for the dataset, we need to fulfil several conditions. In shorthand, the dataset has to be from the eCommerce domain, with product descriptions containing the compatibility information, and fulfil the requirements for the natural language processing tasks. I describe the conditions in-depth in this part of the thesis.

Finally, the last part of this thesis focuses on the machine learning model that processes the dataset and identifies compatible products. We focus on leveraging pre-trained models that could be beneficial in this task. The final model consists of several submodels that solve the tasks examined in the previous part.

Chapter 2

Natural Language Proccessing

"Natural language processing (NLP) is the subfield of computer science concerned with using computational techniques to learn, understand, and produce human language content." [1] The field itself is vast, with many practical applications. I will describe in detail only a tiny part of the field. Nevertheless, there are numerous subtasks and challenges. In general, we can categorize them by their application: [2]

- Indexing and searching large texts,
- Information retrieval,
- Classification of text into categories,
- Information extraction,
- Automatic language translation,
- Automatic summarization of texts,
- Question-answering,
- Knowledge acquisition,
- Text generations/dialogues.

The most helpful subtask for this thesis is Information extraction (IE). We need to extract the data in a specific structure that prioritizes the compatibility of the two products.

2.1 Information Extraction

As the name "Information Extraction" (IE) suggests, the task aims to analyze documents and extract domain-specific information in a specified structure. "The general architecture of an IE system is 'a cascade of transducers or modules such that, at each step structure is added to the document and, sometimes, it filters relevant information by application of rules.'" [2]

The transducers and modules can be represented by separate tasks. These are the most notable tasks: [3]

- Parts-of-Speech tagging – tagging words with Parts-of-Speech labels such as nouns, adjectives and verbs.
- Parsing – syntactic analysis of sentences, analysis of syntactic dependencies between words.

- **Named Entity Recognition** – recognizing important entities, e.g. Person, Organization, Location.
- Named Entity Linking – linking recognized entities to entries in a knowledge base.
- Coreference Resolution – identifies words and phrases that refer to the same entities.
- Temporal IE (Event Extraction) – identification of events (who did what to whom, where, when and why).
- **Relationship Extraction** – detection of relationships between identified entities.
- Knowledge Base Reasoning and Completion – the final task of IE – building a structured database of entries (knowledge base).

I highlighted tasks described in more detail in the following sections, as they are relevant to this thesis.

2.2 Named Entity Recognition

Named entity recognition (NER) is a subtask of NLP that focuses on IE. There are several definitions of named entities (NE). The most notable ones are:

> "The Named Entity task consists of three subtasks (entity names, temporal expressions, number expressions). The expressions to be annotated are "unique identifiers" of entities (organizations, persons, locations), times (dates, times), and quantities (monetary values, percentages)." [4]

> "Named entities are phrases that contain the names of persons, organizations and locations." [5]

> Alternatively, as defined in [6], there are several recognized NEs in their task: Amount, Facility, Geo-Political Entity, Localization, Organization, Person, Product, Time, and Unknown.

I will work with a more general definition of NE in this thesis. NEs are entities referring to specific information relevant to the given domain. Therefore, my first task in the practical part of this thesis will be to define the named entities to be recognized. In short, the NEs in this thesis will capture vital information identifying the products.

The main task is to extract these NEs from the documents (Figure 2.1). The task is performed by learning to recognize NEs using a labelled dataset.

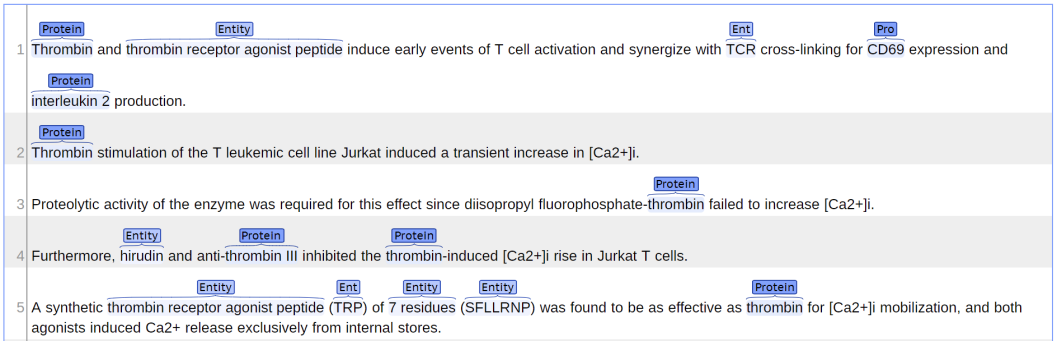

Figure 2.1 Example of identified NEs. Adapted from examples in Brat software. [7]

2.3 Relationship Extraction

"Relationship extraction (REL) is one of the key tasks involved in information extraction. It refers to classification of semantic relationship that can exist between entities." [8] The REL task is applied after the NER task. After identifying the NEs, the model will identify the relationships between them. The most crucial relationship will be the compatibility of the two products. I will need to define these relationships right after specifying the NEs. This task will need additional annotations in the dataset (Figure 2.2).

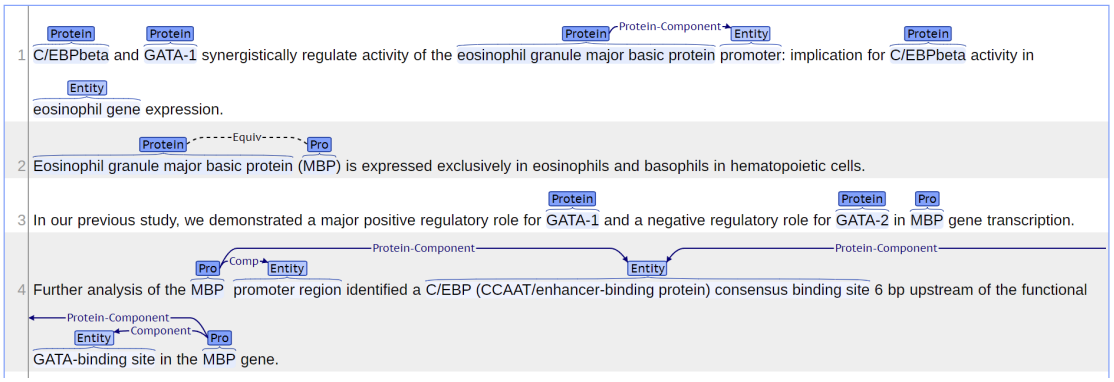

Figure 2.2 Example of REL annotated document. Adapted from examples in Brat software. [7]

2.4 NLP approaches

To implement the outlined tasks, I will need to first explain the standard techniques in NLP.

2.4.1 Word embeddings

The first step in NLP tasks is to map the words into embeddings. We could divide the embeddings into two categories. First, there are context-independent embeddings such as word2vec. This model embeds each word as a vector of decimals. Embeddings produced with word2vec comply with simple algebraic operations. The most famous example is $vec(King) - vec(Man) - vec(Woman)$ is closest to the $vec(Queen)$ representation. [9]

The second type is context-sensitive embeddings. The produced embeddings are vectors of decimals as well. There can be various approaches to defining what context is considered within each embedding. For example, the context can be formed by processing several preceding words or taking into account the context of the whole sentence. However, one word will have different embeddings because the context will affect the final values.

2.4.2 Recurrent Neural Networks

Traditional neural networks are not applicable for a series of data; they can learn the context-independent embeddings. To learn context-sensitive embedding, we need to use at least a recurrent neural network (RNN). The simple idea behind RNNs is to pass the previous output and current word as input, as depicted in Figure 2.3.

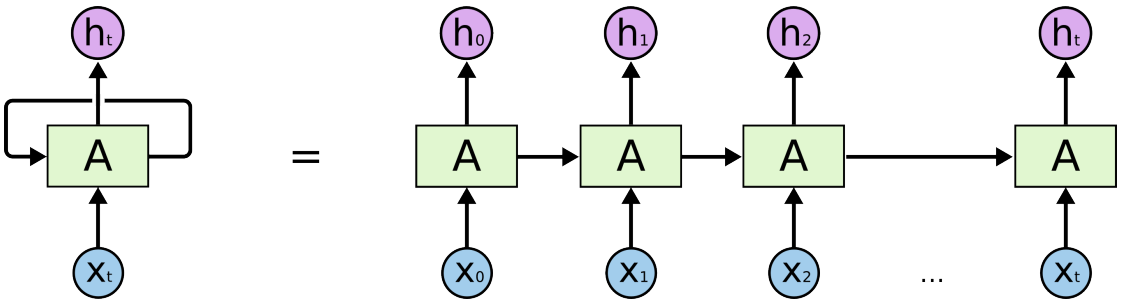

Figure 2.3 Recurrent neural network and unrolled recurrent neural network side by side. [10]

2.4.3 Long short-term memory

Long short-term memory (LSTM) is a more complex RNN introduced by [11]. As the name suggests, the model takes into account short term and long term context. It is done by introducing a cell state updated with a special change operation, capturing the long term context. In Figure 2.4, the cell state is depicted by the upper horizontal line.

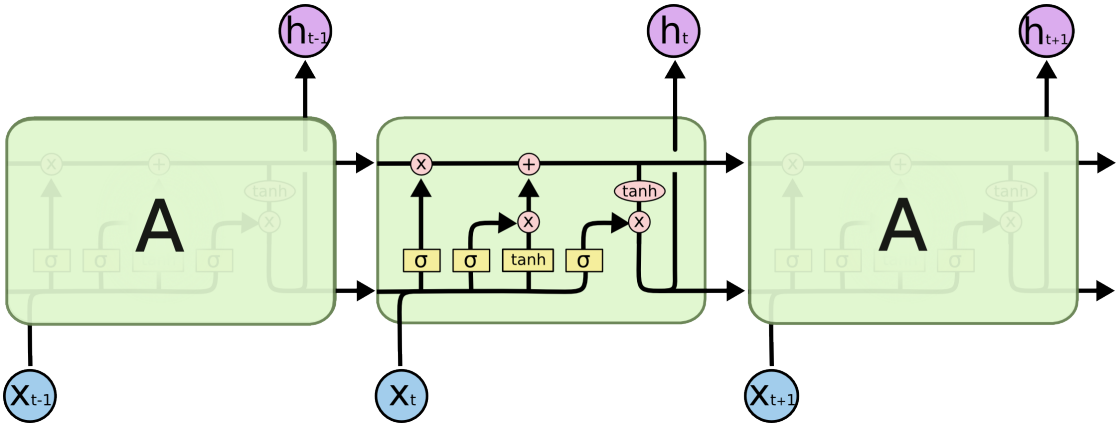

Figure 2.4 Unrolled LSTM model with four interacting layers. [10]

The LSTM enables us to carry context in the forward direction. If you want to leverage context from both directions, you can introduce a backward recurrent LSTM network. This type of bidirectional LSTM was used in many models; one of them is the ELMo model [12], simplified version of the ELMo model is in Figure 2.5. ELMo model was the state of the art model (at the time) in several tasks.

ELMo pre-trained model is published under Apache License, version 2.0; the model is available for usage and modification. I could use the ELMo model as part of my final model.

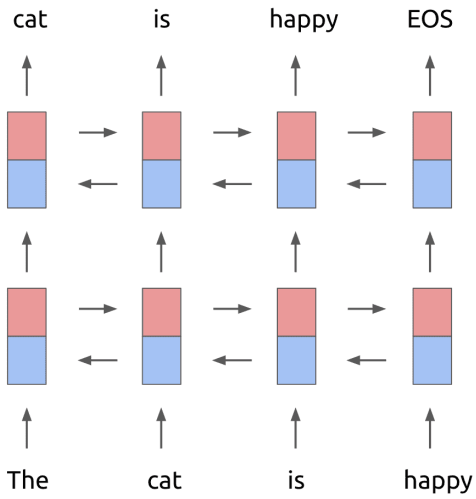

Figure 2.5 ELMo model uses forward (red boxes) and backward (blue boxes) recurrent LSTM. [13]

2.4.4 Attention

In NLP tasks, it is useful to introduce weights for each word based on its relevance to the task. The attention mechanism solves this problem by learning to predict the relevance of input elements. [14] The mechanism learns to predict the importance of input elements for the output of the given task. "Self-attention, sometimes called intra-attention is an attention mechanism relating different positions of a single sequence in order to compute a representation of the sequence." [15] An example of self-attention is displayed in Figure 2.6.

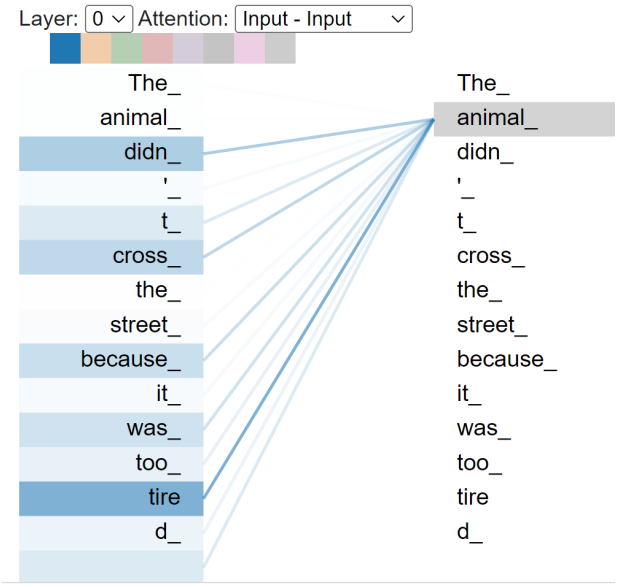

Figure 2.6 Visualization of self-attention for one word, using tensor2tensor visualization.[16]

2.4.5 Transformer

Recurrent models described in previous sections have simple architecture, but we need to process the document one word at a time, and the task is not parallelizable. For the introduced models, the number of operations grows linearly with the length of the document. Moreover, the further apart are two related elements, the more difficult it is not to forget their context when processing the document. "The Transformer model architecture eschews recurrence, and instead relies entirely on an attention mechanism, to draw global dependencies between input and output." [15] The model follows encoder-decoder architecture as depicted in Figure 2.7.

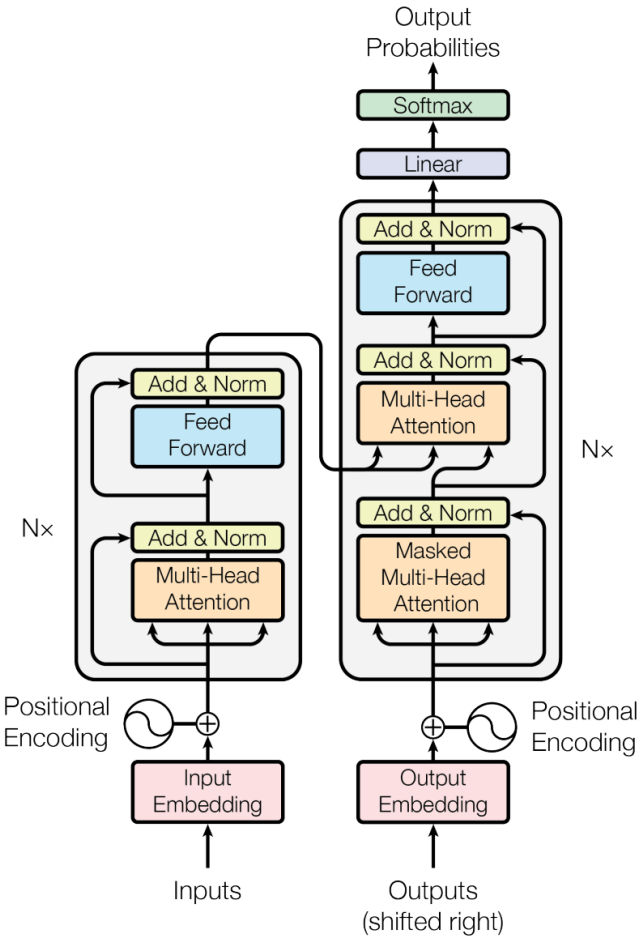

Figure 2.7 The Transformer model architecture, the left column corresponds to the encoder and the right to the decoder. [15]

The model calculates the attention for several input-output combinations: input self-attention (input-input), output self-attention (output-output) and input-output attention. The attention calculation is done using a multi-head attention module consisting of multiple scaled dot-product attention modules; both are depicted in Figure 2.8

The identical model architecture was trained as the BERT model [17] for language understanding; the model was trained to be applicable in several NLP tasks. "The masked language model randomly masks some of the tokens from the input, and the objective is to predict the original vocabulary id of the masked word based only on its context." [17] This approach pre-trained the model for usage with token-level tasks, such as NER or REL.

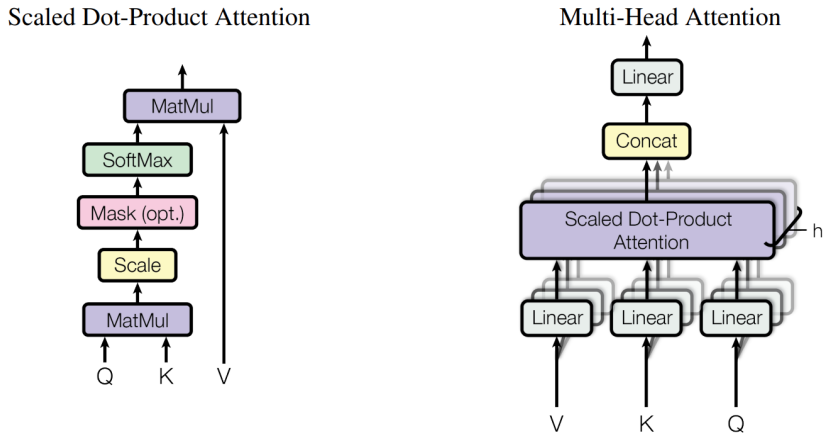

Figure 2.8 The Transformer model architecture, the left column corresponds to the encoder and the right to the decoder. [15]

The model was further optimized and trained for a longer time on a ten times larger dataset and published as RoBERTa. [18] Both BERT and RoBERTa are published as open-source models. BERT model is published under Apache License, version 2.0, and RoBERTa model is published under GNU General Public License, version 2. I chose to use RoBERTa as the pre-trained model for fine-tuning for NER and REL tasks.

Chapter 3

Preparing the dataset

There are several conditions we need to meet when searching for the dataset. Firstly there are restrictions concerning the domain of the contents of the dataset. The documents inside the dataset need to be from the eCommerce domain – containing the products' names and descriptions. The products have to fall into a category where it is possible to identify compatibility.

For example, if the products were from the books category, there would be no compatibility information. You could buy two books, and it would not matter which book you read first. Alternatively, you can buy only one and fully enjoy reading it. Even several years after buying it, you can read it without replacing parts of the book or without any obstacles.

On the other hand, you can run the printer right out of the box when shopping for inkjet printers, but when using the printer regularly, you will need to replace the ink sooner or later. If you use the printer once and then use it for the second time after a couple of years, the ink could be dried out, and you would need to replace it anyway. Alternatively, other problems could arise, and you would need to replace some parts of the printer, e.g. cables or printhead.

Secondly, all the documents in the dataset need to be written in the English language. Each document should contain data for exactly one product. The minimum contents for each document are the product's name and description. There can also be other valuable data, such as the category of the product. If the dataset contained images of the products, I would not be processing those because this thesis focuses on NLP tasks.

Lastly, there are requirements for the NER and REL tasks. Besides the description and name, we also need annotations for the documents. The NER annotations are in the form of labels assigned to one or more words. Each product mentioned in the document needs to have a label assigned. The model is indifferent to the label names as long as the annotation style is consistent for all the texts. The REL annotations connect the NER annotations with an assigned label specifying the type of relationship. The relationship label that is essential for this thesis is the compatibility one.

In summary, these are the conditions for the dataset:

1. The dataset has to be in the English language and from the eCommerce domain.

2. Documents in the dataset have to contain the name and description of one product.

3. The dataset has to contain annotations for NER and REL tasks.

If I manage to find a dataset that partially fulfils the conditions, I can still take the dataset and add the missing part. For example, if only REL annotations were missing but other conditions were fulfilled, adding the annotations would not be very difficult.

3.1 Examined datasets

I examined several datasets; I used two approaches when looking for the dataset. Either I found the dataset by looking for task-specific datasets or by looking for the datasets in eCommerce. These are some of the datasets I found and examined.

BioNLP [19]

This dataset is part of the BioNLP shared task workshop that focuses on IE tasks in the biomedical domain. The dataset contains all the necessary annotations for the NER and REL tasks.

I cannot use this dataset since it is from the domain of biomedicine and its main focus is on constructing a knowledge base about specific proteins.

ACE 2005 [20]

The dataset contains broadcast news, newswire, discussion forums, and other documents in English, Chinese and Arabic. There are annotations for several NER labels: Person, Organization, Location, Vehicle... The dataset contains REL annotations; these link the NEs with the relation, characterization or event detection.

The dataset is not suitable; it is not in eCommerce and does not contain compatibility information.

SCIERC [21]

One document in this dataset corresponds to an abstract of one scientific article. The document contains NER and REL annotations. Following NEs are used in this dataset: method, task, metric and other NEs. The NEs are linked by relationships such as used-for, part-of, and coref (as in coreference).

The dataset is not in the eCommerce domain; therefore, I cannot use this dataset.

The Klarna Product Page Dataset [22]

The dataset contains HTML of product pages from various eCommerce websites. There are annotations for HTML elements containing price, name, main picture, add to cart and cart. This dataset does not contain any NER or REL annotations.

Since the dataset is in the right domain, it could be used for the final dataset. I would need to extract the product names and descriptions and create the NER and REL annotations.

Amazon product data [23]

This dataset contains Amazon product data. There are two main parts of the dataset, metadata (containing the title, descriptions, price, categories...) and reviews (containing the reviewer name, rating, review text, review time...). The dataset does not include any NER or REL annotations.

This dataset is in the right domain and is more suitable than the Klara product page dataset because it contains all the necessary texts in structured data. However, I would still need to prepare the NER and REL annotations.

Finer [24]

This dataset contains technology news articles, with some product-specific articles. There are annotations for the following NEs: organization, location, person, product, event and date. There are no REL annotations, and the articles are written in Finnish.

The dataset is not suitable as it is not in English. Even though the dataset is not from the right domain, some of the documents might have been useful as there are the product NEs, if the dataset was in English.

3.2 Creating own dataset

I have found the Klarna product page dataset and Amazon product data dataset, both of them are from the right domain and do not contain any NER and REL annotations. After considering both options with my supervisor, we decided to create our dataset. The main reason is that we can extract a sufficient number of product data from a more narrow product category. We decided to create the dataset from the printer, printer cartridges and printer accessories categories.

3.2.1 Data scraping

I prepared a python script for scraping the data from Amazon using the python library Beautiful Soup [25] for extracting the data from downloaded HTML documents. For each product in the selected categories, I extracted the following texts: title, category hierarchy, bullet-point description, and the product page URL. All of the extracted information is depicted in Figure 3.1.

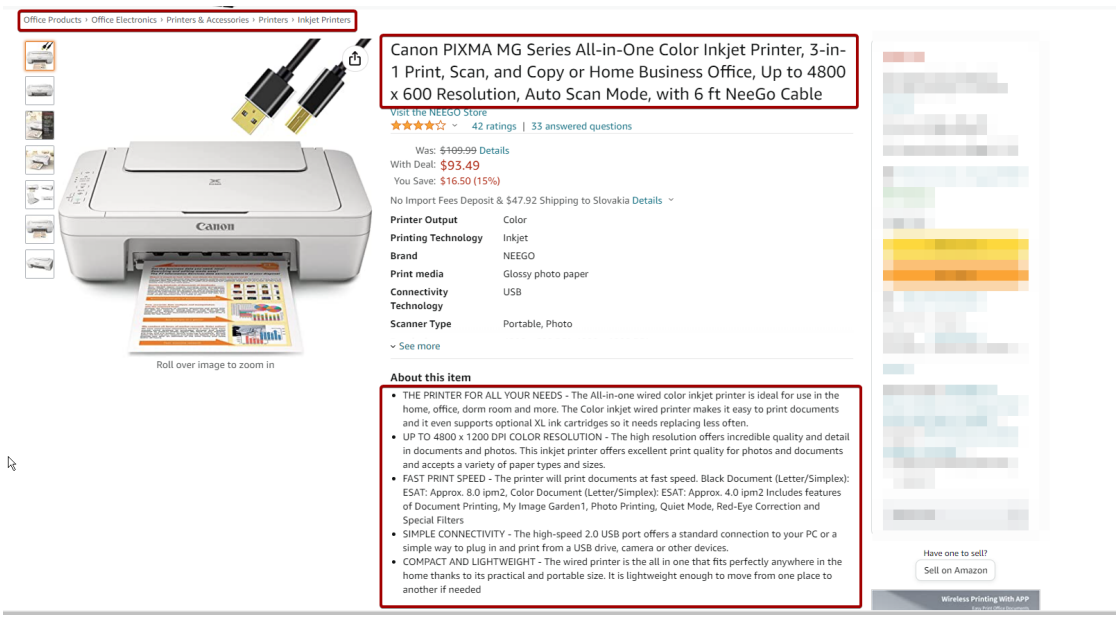

Figure 3.1 Scraped information from Amazon.com product page. [26]

The extracted information is concatenated into one text document in the following order: title, URL, category, and bullet-point description. A sample document after concatenation can be found in Figure 3.2.

```
Brother Wireless All-in-One Inkjet Printer, MFC-J491DW, Multi-function
Color Printer, Duplex Printing, Mobile Printing,Amazon Dash
Replenishment Enabled, Black, 8.5 (MFCJ491DW)
https://www.amazon.com/Brother-Wireless-MFC-J491DW-Multi-Function-Repl
enishment/dp/B07C4V4WWF/ref=sr_1_86
Office Products > Office Electronics > Printers & Accessories
> Printers > Inkjet Printers

Make sure this fitsby entering your model number.
Simple to connect:  Choose from built in wireless or connect locally to
a single PC or Mac via USB interface.  MFCJ491DW offers easy to set up
wireless networking
Mobile printing:  Print wirelessly from mobile devices(1) using Air
Print, Google Cloud Print, Brother iPrint & Scan, Mopria and Wi Fi
Direct
Cloud connectivity:  Scan to popular Cloud services directly from the
printer including Google Drive, Dropbox, Box, One Drive and more(2)
Versatile paper handling:  Automatic document feeder and up to 100
sheet capacity paper tray for letter/legal size paper for flexible
printing
For use with brother genuine inks:  LC3011BK, LC3011C, LC3011M,
LC3011Y, LC3013BK, LC3013C, LC3013M, LC3013Y
Amazon dash replenishment enabled:  Upon activation, Amazon Dash
Replenishment measures the ink level and orders more from Amazon when
it's low
```

Figure 3.2 Sample document after concatenation of extracted information.

I followed the pagination in the list of products and successfully scraped more than 3600 product pages from Amazon's category: *Office Products – Office Electronics – Printers & Accessories*.

However, since I tried downloading all products in the given category, I downloaded duplicate products and product variants in different colours/configurations. I describe the code used in chapter 5.1.

3.2.2 NER annotations

As described in section 2.2, the first task is to choose which NEs will capture the information identifying the products. Upon inspecting the texts extracted from product pages, it became clear that there are rules that the product names follow. Most of the printers follow the rule of having three identifying information. The most general identifier is the brand of the product, such as *Canon, HP, Brother...* Then there is the identifier of the product line, e.g. for HP those are *DeskJet, LaserJet, OfficeJet...* Finally, there is the identifier of the specific product, the product code: *1000, 2050, 3510...*

These three partial identifiers can be right next to each other, or other words can separate them. For example, the official name of the product can be *Canon PIXMA MG2525*, but the product page title can be more complex: *Canon MG Series PIXMA MG2525...* For this reason, I chose to annotate the three parts of the product name separately, with the labels: Brand, Line and Code.

However, this approach did not cover all the products. Some products follow different naming rules; the product names might be more general. For instance, printer storage bags or printhead

cleaning kits do not have a product line and code. They are named in the following format: Brand Printer Storage Bag or Brand Printhead Cleaning Kit. For these types of names, I introduce the fourth label: Product. This label should be used for more general names. The usage of both names can be viewed in Figure 3.3.

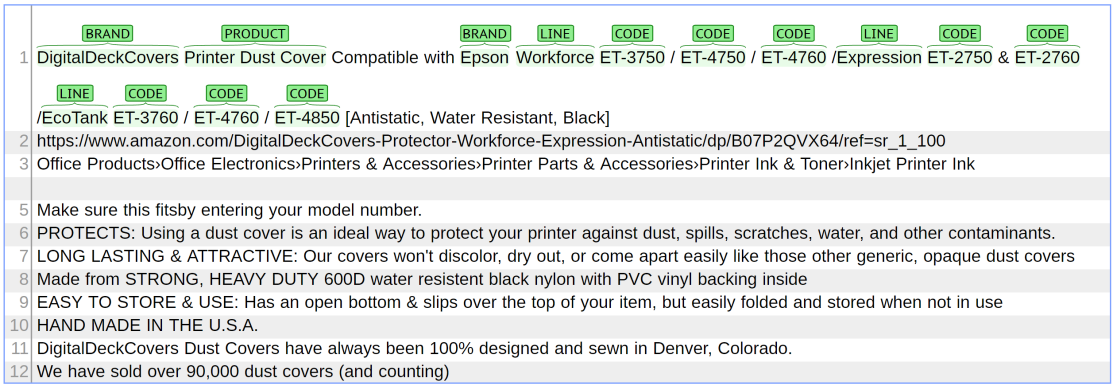

Figure 3.3 Usage of NER labels in a document. Image created from Brat software. [7]

In summary, I chose the following NEs to be annotated for the NER task:

- Brand – brand of the product, e.g. HP, Epson, Brother.
- Line – product line, e.g. LaserJet, LaserJet Pro, DeskJet.
- Code – product code, e.g. 1010, 1020, 4010.
- Product – general product name, if the product cannot be defined by product line and code, e.g. printer carrying case,

Each product should be defined by NEs: Brand-Line-Code or Brand-Product.

3.2.3 REL annotations

Now I need to connect the NEs with relationships. There are two primary relationships between NEs.

The first type connects identifiers for one product. I divided it into two separate relationships. The first relationship is between NEs Brand and Line; this relationship will be annotated with the label From. The second relationship is between NEs Brand and Product, or Line and Code. I will annotate this relationship with the label Of.

The second type of relationship denotes the compatibility of two different products. For this relationship, I will be linking just the most granular NEs identifying the products, Code and Product NEs. The relationship can connect any combination of the two NEs and annotate the Compatible label. An example of an annotated document can be found in Figure 3.4.

In summary, I chose the following relationships to be annotated for the REL task:

- From – *Line-from-Brand*.
- Of – *Code-of-Line*, or *Product-of-Brand*.
- Compatible – *(Code/Product)-compatible-(Code/Product)*.

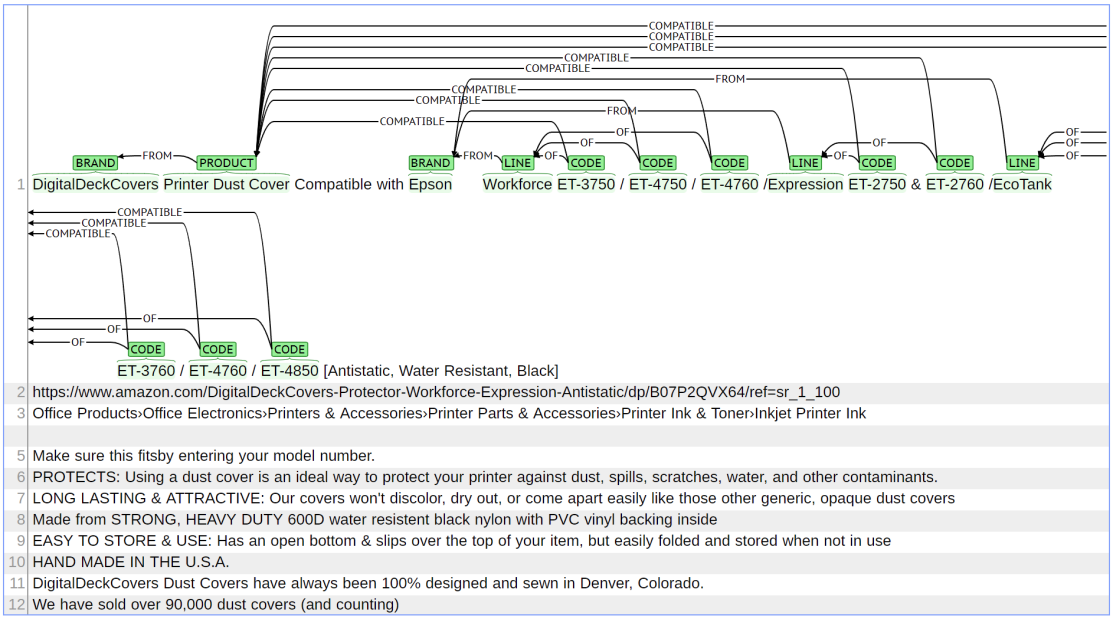

Figure 3.4 Usage of NER and REL labels in a document. Image created from Brat software. [7]

3.2.4 Annotation tool

When choosing the annotation tool, there were not so many options. I looked primarily for an open-source project I could run from a Docker image. Initially, I annotated the documents with the open-source software Doccano[1]. This software was sufficient for NER annotations. Even though the user interface seems to be prepared for REL annotations, the functionality is not implemented yet.

The second choice was open-source software Brat[2]. This software is capable of annotating both NER and REL annotations. It is required to save each document from the dataset in a separate text file; the annotations are saved in a separate text file with the *.ann* extension. An example of the annotation file can be found in Figure 3.5.

```
T1 BRAND 0 2 HP
T2 LINE 3 16 OfficeJet Pro
T3 CODE 17 22 9015e
R1 OF Arg1:T3 Arg2:T2
R2 FROM Arg1:T2 Arg2:T1
```

Figure 3.5 File containing annotations. Created by Brat software. [7]

3.2.5 The dataset

Using the described set-up, I annotated 121 documents, and only 64 out of those have at least one relationship with the label *COMPATIBLE*. These documents will be used for training, validating and testing the model.

[1] https://doccano.github.io/doccano/
[2] https://brat.nlplab.org/

3.2.6 Expanding dataset

Since I have only 121 annotated documents, I decided to expand the dataset by generating new documents from the existing ones. To fulfil this task, I wrote additional python code that implements the following algorithm:

1. Identify documents with a large number of compatibility relationships.
2. For each identified document:
   a. Identify NEs that are compatible with only one other NE.
   b. While generating documents:
      i. Generate a subset of NEs with a single compatibility relationship.
      ii. Delete all other NEs with a single compatibility relationship.
      iii. Fix annotations.
   c. Collect generated documents.1
3. Return generated documents.

An example of the generated document is displayed in Figure 3.6. The generated document has room for improvement, e.g. removing excess commas.

```
Original HP 61 Black Ink Cartridge | Works with DeskJet 1000 , 1010
, 1050 , 1510 , 2050 , 2510 , 2540 , 3000 , 3050 , 3510 ; ENVY 4500
, 5530 ; OfficeJet 2620 , 4630 Series | Eligible for Instant Ink |
CH561WN
https://www.amazon.com/HP-Black-Cartridge-CH561WN-Deskjet/dp/B003H2GBM
4/ref=sr_1_92
Office Products>Office Electronics>Printers & Accessories>Printer Parts
& Accessories>Printer Ink & Toner>Inkjet Printer Ink

Original HP Ink is engineered to work with HP printers to provide
consistent quality , reliability and value
This cartridge works with :  HP DeskJet 1000 , 1010 , 1050 , , , , 1510
, 1512 , 2050 , 2510 , , , 2540 , , , , , , , , , 3000 , 3050 , , , , ,
, , 3510 , ,
This cartridge works with :  HP ENVY 4500 , 4501 , , , , 5530 , , , , ;
HP OfficeJet 2620 , 4630 , 4632,4635
Cartridge yield ( approx . ):  190 page
Up to 2x more prints with Original HP Ink vs.  non - Original HP Ink
82 % of HP ink cartridges are manufactured with recycled plastic
Get ink your way :  buy Original HP Ink Cartridges or save up to 50 %
and never run out with HP Instant Ink , the Smart Ink Subscription
What 's in the box :  1 new Original HP 61 Black Ink Cartridge (
CH561WN )
```

Figure 3.6 Document generated after deleting subset of NEs with single compatibility relationship.

After adding the generated documents, the dataset comes close to 1000 documents.

Chapter 4
Model architecture

For building the project, I decided to leverage the spaCy library[1]. This library specializes in building and training NLP models. SpaCy models can be initialized with pre-trained models such as RoBERTa or BERT. The whole spaCy model is described as a pipeline built from components. Each component can be initialized with previous custom trained models, pre-trained models, or trained from scratch. A diagram of the spaCy pipeline can be found in Figure 4.1

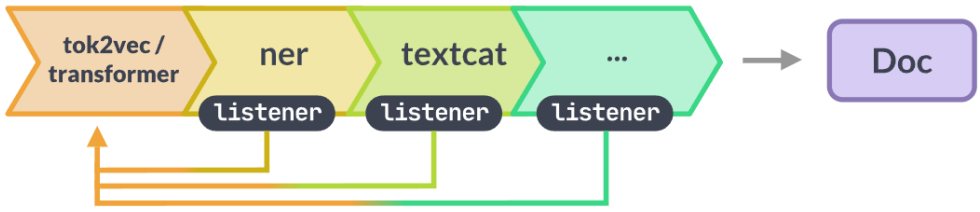

Figure 4.1 Spacy pipeline. Adapted from spaCy usage documentation. [27]

Each component has its model definition and pipe definition. The model represents the model with all its weights and provides an interface for computing the forward result with a backpropagation callback. Trainable pipe implements all methods necessary to process documents and interact with the model, e.g. predict, update, get_loss and score. The spaCy model is set up using a configuration file, where all hyperparameters are set up. The only condition is that the dataset must be saved in a proprietary binary file with a *.spacy* file extension.

The library and whole framework are easy to set up; therefore, they are easily replicable. Moreover, the library is open-source software published under an MIT license. These factors contribute to the overall transferability of the trained models.

4.1 NER model

The NER model uses two components. The first is the Transformer model, which is initialized with a pre-trained RoBERTa model. In the configuration file, we can set the length of the window (number of tokens processed at one step) and stride (the distance the window moves each step) for the Transformer model.

[1] `https://spacy.io/`

The second component is the entity recognizer. This component is prepared in spaCy natively. It is a transition-based NER component. "It is a transition-based named entity recognition component. The entity recognizer identifies non-overlapping labelled spans of tokens." [28] The loss function does not take into consideration partially matched NEs.

A detailed description of the NER model configuration file can be found in Appendix A.

4.2 REL model

The REL model uses two components as well. The first component is the Transformer model, which was already finetuned for the NER task. The second component is the relationship extractor; there is no natively implemented component for this task in the spaCy library. Therefore I have to use custom component implementation. I will adapt the REL component from spaCy projects. [29]

The REL model consists of one linear layer with a logistic activation function. The component takes input pairs of NEs in their vector representations (produced by the transformer) and predicts their probability for every relation label. The REL component was further optimized to implement customizations and further functionality; this is discussed in the next chapter.

Chapter 5
Implementation

There are several scripts and custom components that need to be implemented. In this chapter, I will describe the relevant algorithms.

5.1 Document scraping

In subsection 3.2.1, I described the data scraping from product details on Amazon.com. The script I implemented for the task is split into several parts.

First, we have the task of getting the URL address for the product pages. It is done by starting from the initial category page, extracting all relevant links and proceeding to the next page. The links that I extract are further stripped of any unnecessary query parameters, which are deduplicated.

The next task was to download the product details. I crawl the deduplicated product links and save the HTML source codes for each product page. Afterwards, I proceed with extracting the relevant parameters of the page.

I use the library Beautiful Soup[1] to extract the links from category pages and product parameters from product pages. Using this library, I can parse one HTML file at a time and use CSS selectors to extract the desired values. All the documents are saved in JSON lines file, and the first 400 documents are saved as individual text files, prepared for annotation.

5.2 Brat to spaCy dataset conversion

There are several challenges when converting the dataset from Brat format to spaCy format. The first of them is different annotating formats. Brat format references all NEs by the first and the last character position, while spaCy uses the first and the last token position. The REL annotations are referenced by the NE identifier generated in Brat, while in spaCy, they are referenced by the first token position of the NE. The script has to cycle through all the annotations for each document and map the character position to token positions. During this process, it is also needed to fix miss-aligned NEs; the miss-alignment happens if whitespace or comma are marked as the last/first character of NE.

Initially, I was predicting just the labelled relationships. However, the initial results were not promising, so I introduced the no relationship label – *NO_REL*; this resulted in one additional step for the conversion script to fill all unlabeled relationships with the *NO_REL* label.

[1] `https://www.crummy.com/software/BeautifulSoup/`

The Second significant update was converting direction-aware relationships to symmetrical ones. If *product A* is *compatible* with *product B*, the relationship has to work in the opposite direction as well. I implemented a new processing step to avoid distinguishing between the directions in which the relationship is annotated. In this case, I decided to transform the relationships so only pairs where the first NE is sooner in the text than the second NE were considered. I updated the script to turn around all incorrectly rotated relationships, and additionally, I had to update the REL model to abide by the same rule.

Finally, I implemented the expansion of the dataset by generating new documents; this is discussed in subsection 3.2.6.

5.3 REL model customizations

When customizing the REL model, I primarily edited the *get_loss* and *_examples_to_truth* functions; refer to figure 5.1. There were two significant additions to the initial model.

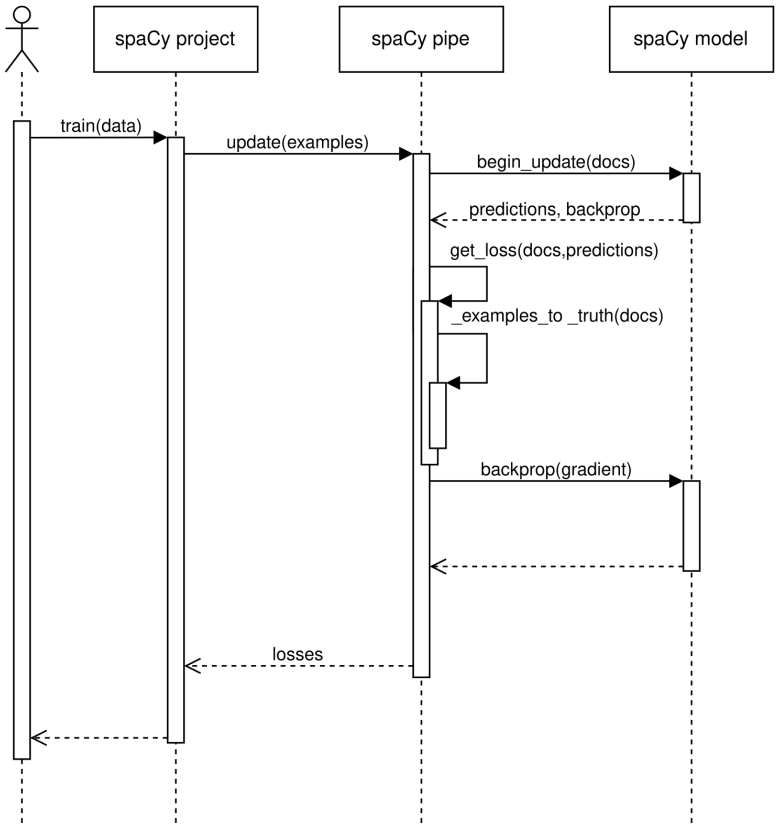

Figure 5.1 Sequence diagram for spaCy training.

The first update was the introduction of weights for the predicted labels; this allowed me to set a lower weight for the *NO_REL* relationship and higher priority for the *COMPATIBLE* relationship. The major update was in the calculation of gradient loss used in backpropagation. This update helped the model focus on more significant relationships, but it was insufficient because the dataset contains around 50 times as many *NO_REL* relationships as *COMPATIBLE* ones. The only change in the *get_loss* function was on line 4, displayed in Figure 5.2, where *self.label_weights* is very simple array of weights.

Therefore I introduced gradient filtration. This update allowed me to clear loss for part of the *NO_REL* relationships. The loss for the *NO_REL* relationship is cleared only when a randomly generated number between 0 and 1 is higher than a threshold. The spaCy library did not enable me to filter out the selected gradients completely; I can only set the gradient to zeros for the selected relationships. The main change in *_examples_to_truth* (Figure 5.3) is on lines 16 to 19, where the filtration matrix is prepared. The matrix is then used in *get_loss* (Figure 5.2) on line 6, where the filter is applied to the gradients.

Both custom weights and threshold for filtration are introduced as hyperparameters for the REL model.

```
1   def get_loss(self, examples: Iterable[Example], scores) -> Tuple[float, float]:
2       truths,filter = self._examples_to_truth(examples)
3       gradient = (scores - truths)
4       gradient=gradient*self.label_weights
5       mean_square_error = (gradient ** 2).sum(axis=1).mean()
6       gradient = gradient*filter
7       return float(mean_square_error), gradient
```

Figure 5.2 Updated get loss function.

```
1   def _examples_to_truth(self, examples: List[Example]) \
2       -> Optional[Tuple[numpy.ndarray,numpy.ndarray]]:
3       nr_instances = 0
4       for eg in examples:
5           nr_instances += len(self.model.attrs["get_instances"](eg.reference))
6       if nr_instances == 0:
7           return None
8       truths = numpy.zeros((nr_instances, len(self.labels)), dtype="f")
9       filter = numpy.ones((nr_instances, len(self.labels)),dtype="f")
10      c = 0
11      for i, eg in enumerate(examples):
12          for (e1, e2) in self.model.attrs["get_instances"](eg.reference):
13              gold_label_dict = eg.reference._.rel.get((e1.start, e2.start), {})
14              for j, label in enumerate(self.labels):
15                  truths[c, j] = gold_label_dict.get(label, 0)
16                  if label=="NO_REL" and gold_label_dict.get(label, 0)==1 \
17                      and nr_instances>10 \
18                      and random.random()>self.no_rel_filter:
19                      filter[c]=numpy.zeros(len(self.labels),dtype="f")
20              c += 1
21      filter = self.model.ops.asarray(filter)
22      truths = self.model.ops.asarray(truths)
23      return (truths,filter)
```

Figure 5.3 Updated examples to truth function.

5.4 Evaluation

I implemented a script with several functions helpful in evaluating the trained model. The script can evaluate the NER model, the REL model, or the NER and REL model combination.

I calculate the score for each label (NEs or relationships) and a total score. I also added advanced functions designed to deduplicate the products and calculate the scores for predicting unique product compatibility. If there are multiple instances of the same product, it is sufficient to predict compatibility for just one of the instances to mark the deduplicated product as compatible.

Chapter 6

Training and evaluation

The model was trained as two separate models: the NER and the REL models. I decided to train them separately to optimize the time needed for training the models. Additionally, when I tried to merge the models into one pipeline, I encountered technical difficulties with the spaCy library. We decided with my supervisor to focus on optimizing the models rather than solving the technical difficulties.

6.1 Scoring function

For scoring the models, I will be using precision and recall as supporting metrics and F1-score as the primary metric.

$$precision = \frac{true\ positives}{true\ positives + false\ positives}$$

$$recall = \frac{true\ positives}{true\ positives + false\ negatives}$$

$$F_1 = 2\frac{precision \cdot recall}{precision + recall}$$

6.2 Dataset bias

The dataset is heavily biased; there are several reasons for the dataset to be biased:

1. The number of annotated documents is not sufficient – around 120.
2. The annotated documents are handpicked from the scraped documents to meet specific criteria. The result of this handpicking is survivorship bias.

I had two options if I wanted to split the dataset into training, testing and validation datasets. I could either split the dataset before the expansion of the dataset or split the dataset afterwards. These approaches would not provide a fair assessment of the trained models. If I split the dataset before the expansion, I would have limited validation and testing datasets of 20 to 30 documents. Moreover, if picked the split entirely randomly, it could happen that one of the datasets would not contain any documents with the *COMPATIBLE* relationships. If I split the dataset after the expansion, it would be very probable that documents generated from one original document would be present in all three parts of the dataset.

Since both options for splitting the dataset are very limited, the final score would probably be heavily biased in both options. I decided to not split the dataset into training, testing and validation datasets and work with just one dataset for all the tasks. It is not the best solution, but it is the best I have available.

6.3 Training NER

The NER model was trained for 20 000 training steps using Google Colab[1] with 16 GB of dedicated GPU memory; it took 4 hours to train the model. The model was trained using the extended dataset.

Model	precision	recall	F1-score
NER	0.9894	0.9535	0.9711

Table 6.1 Results for the NER model.

After the initial training, the model seemed promising (Table 6.1); there were no changes to the NER model architecture. The reported F1-score is very high, most probably due to overtraining. Tests of the trained model on unlabeled data show the model can identify NEs for manufacturers whose products were annotated in multiple documents. The model cannot recognize products from manufacturers not represented in the training dataset; this is expected behaviour; an example of this happening is present in Figure 6.1.

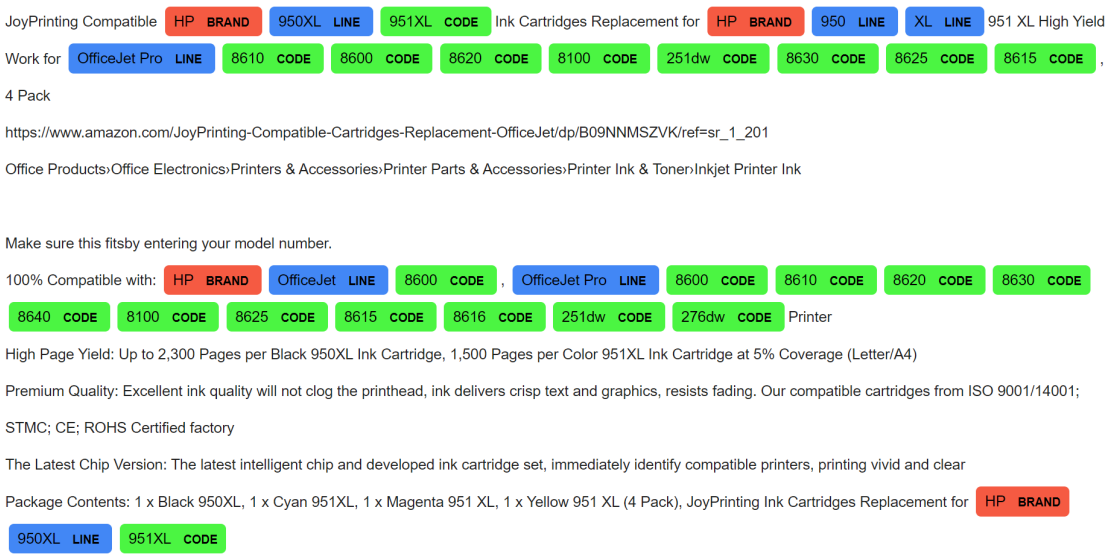

Figure 6.1 Prediction of the NER model on a previously unseen document.

[1] https://colab.research.google.com/

6.4 Training REL

The REL model was trained for 20 000 training steps on a graphics card with 8 GB of dedicated memory; the average training time was around 10 hours. The model was trained using the extended dataset.

Initially, the model performed poorly, with an F1-score less than 5%. The first significant improvement was after introducing the *NO_REL* relationship; it doubled the F1-score to a score of just under 10%. The results are calculated only for *FROM, OF,* and *COMPATIBLE* relationships. I did not include these two scores in the table of scores for the REL model.

The second improvement was after adding weights for the relationships; the results are in Table 6.2. The weights are stated in the following order: *FROM, OF, COMPATIBLE, NO_REL*.

model	weights	precision	recall	F1-score
REL	2,2,5,0.1	0.0922	0.1552	0.1157

Table 6.2 Results for the REL model after the introduction of weights.

The following update doubled the score, changing from direction-aware relationships to symmetrical relationships; results are shown in Table 6.3.

model	weights	precision	recall	F1-score
REL - symm.	2,2,10,0.02	0.2369	0.2499	0.2432

Table 6.3 Results for the REL model after the change to symmetrical relationships.

Implementation of filtration for the *NO_REL* relationship (more details on implementation in section 5.3) did not improve the model significantly; results are shown in Table 6.4.

model	filtration	weights	precision	recall	F1-score
REL - symm.	10%	2,2,10,0.02	0.3035	0.2613	**0.2808**
REL - symm.	15%	2,2,10,0.02	0.2170	0.2818	0.2452
REL - symm.	25%	2,2,10,0.02	0.2132	0.1238	0.1566
REL - symm.	90%	2,2,10,0.02	0.1825	0.2550	0.2128
REL - symm.	15%	10,10,10,0.01	0.2235	0.2861	0.2510

Table 6.4 Results for the REL model after the introduction of filtration.

After all the modifications were implemented, I revisited the dataset expansion script. I tweaked the logic of generating new documents, and with the new dataset, I managed to double the F1-score once more; the results are in Table 6.5.

model	filtration	weights	precision	recall	F1-score
REL - symm., new dataset	15%	5,5,10,0.02	0.5191	0.5098	0.5144
REL - symm., new dataset	15%	10,10,10,0.01	0.4692	0.6048	**0.5284**

Table 6.5 Results for the REL model after improving the script for generating the extended dataset.

6.5 Summary

The main task of this thesis was to identify the *COMPATIBLE* relationship; therefore, I decided to calculate the F1-score just for this relationship.

model	filtration	weights	NER score	REL score	score *COMPATIBLE*
direction-aware	-	2,2,5,0.1	0.9711	0.1157	0.2645
symm.	-	2,2,10,0.02	0.9711	0.2432	0.2289
symm.	10%	2,2,10,0.02	0.9711	0.2808	0.2567
symm.	15%	2,2,10,0.02	0.9711	0.2452	0.2436
symm.	25%	2,2,10,0.02	0.9711	0.1566	0.2347
symm.	90%	2,2,10,0.02	0.9711	0.2128	0.2561
symm.	15%	10,10,10,0.01	0.9711	0.2510	0.2232
symm., new dataset	15%	5,5,10,0.02	0.9711	0.5144	**0.6230**
symm., new dataset	15%	10,10,10,0.01	0.9711	**0.5284**	0.5329

Table 6.6 Results for trained models with scores for the *COMPATIBLE* relationship. The weights are stated in the following order: *FROM, OF, COMPATIBLE, NO_REL*.

The trained model successfully identified the compatibility with a score of 62.30%; more detailed results are shown in Table 6.6.

Chapter 7
Conclusion

The recent rise of eCommerce brought new challenges to the customers. One of these challenges for the customers is researching the compatibility between various products. In this thesis, I explored solving the problem by training a machine learning model capable of product compatibility extraction from product descriptions.

The main objective of this thesis was to solve the task of product compatibility extraction from freely written product descriptions using a machine learning model. The model should utilise existing models and approaches in the natural language processing field, mainly its tasks: named entity recognition, relationship extraction, and question answering. I introduced the reader to these tasks in Chapter 2 of this thesis.

In the next step, I tried searching for the dataset to train the model. I found some candidates suitable for training named entity recognition and relationship extraction models, but the datasets were in different domains. Therefore, I have created a custom dataset for the model by downloading product titles and short descriptions from Amazon.com. The dataset contains product descriptions for printers, cartridges, and other printer accessories. I described the process of dataset exploration and creation in Chapter 3.

The next step was to prepare annotations for this dataset. I have annotated the dataset using open-source software Brat. The annotations contain named entities and relationships between the named entities. I used four named entities: Brand, Line (product line), Code (product code) and Product; and four different relationship labels between the named entities: from, of, compatible and no_rel. I annotated around 120 product descriptions in this way.

Finally, I implemented the model's architecture, leveraging already pre-trained models. I chose to build the model utilising the Spacy library, allowing me to use pre-trained models or re-use parts of other models easily. The final model uses the pre-trained transformer model called RoBERTa with a combination of models focused on named entity recognition and relationship extraction. The model extracted the named entities with a 97.11% F-score. The relationship extraction model was not as successful and managed to extract relations with a 51.44% F-score. Furthermore, the F-score for compatibility relationship was 62.30%. I explain the model's architecture, implementation, testing and evaluation in Chapters 4, 5 and 6.

After discussion with my supervisor, we have decided to omit the third chosen field – questions answering. There are two reasons for this decision. Firstly, including the questions answering task would result either in the thesis covering more than the scope of a bachelor thesis or having covered all topics only superficially. Secondly, using named entity recognition and relationship extraction is sufficient for fulfilling the main objective, extraction of product compatibility.

Future steps can explore several directions. The first approach would be to improve the relationship extraction component. The second approach would be exploring the application of question answering models; this approach would require modifying the underlying dataset. The

third approach could be testing other architectures and pre-trained models to solve the named entity recognition and relationship extraction. The last approach could explore training more general models that could identify the compatibility of products in different product categories, not just on printers and cartridges; this would require expanding the dataset.

Bibliography

1. HIRSCHBERG, Julia; MANNING, Christopher D. Advances in natural language processing. *Science* [online]. 2015, vol. 349, no. 6245, pp. 261–266 [visited on 2022-03-22]. Available from DOI: 10.1126/science.aaa8685.
2. CHOWDHARY, K. R. Natural Language Processing. In: *Fundamentals of Artificial Intelligence* [online]. New Delhi: Springer India, 2020, pp. 603–649 [visited on 2022-03-22]. ISBN 978-81-322-3972-7. Available from DOI: 10.1007/978-81-322-3972-7_19.
3. SINGH, Sonit. Natural language processing for information extraction. *arXiv preprint arXiv:1807.02383* [online]. 2018 [visited on 2022-03-22]. Available from DOI: 10.48550/ARXIV.1807.02383.
4. MUC6 '95: Proceedings of the 6th Conference on Message Understanding. In: [online]. Columbia, Maryland: Association for Computational Linguistics, 1995, p. 321 [visited on 2022-04-24]. ISBN 1558604022. Available from: https://aclanthology.org/volumes/M95-1/.
5. TJONG KIM SANG, Erik F.; DE MEULDER, Fien. Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition. In: *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003* [online]. 2003, pp. 142–147 [visited on 2022-03-22]. Available from: https://aclanthology.org/W03-0419.
6. GALLIANO, Sebastian; GEOFFROIS, E.; GRAVIER, G.; BONASTRE, Jean-François; MOSTEFA, Djamel; CHOUKRI, Khalid. Corpus description of the ESTER Evaluation Campaign for the Rich Transcription of French Broadcast News. In: *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC'06)* [online]. Genoa, Italy: European Language Resources Association (ELRA), 2006, pp. 140–141 [visited on 2022-03-22]. Available from: https://aclanthology.org/volumes/L06-1/.
7. STENETORP, Pontus; PYYSALO, Sampo; TOPIĆ, Goran; OHTA, Tomoko; ANANIADOU, Sophia; TSUJII, Jun'ichi. brat: a Web-based Tool for NLP-Assisted Text Annotation. In: *Proceedings of the Demonstrations Session at EACL 2012* [online]. Avignon, France: Association for Computational Linguistics, 2012 [visited on 2022-01-14]. Available from: https://brat.nlplab.org/.
8. NASAR, Zara; JAFFRY, Syed Waqar; MALIK, Muhammad Kamran. Named Entity Recognition and Relation Extraction: State-of-the-Art. *ACM Comput. Surv.* [Online]. 2021, vol. 54, no. 1, p. 39 [visited on 2022-04-24]. ISSN 0360-0300. Available from DOI: 10.1145/3445965.
9. MIKOLOV, Tomas; CHEN, Kai; CORRADO, Greg; DEAN, Jeffrey. *Efficient Estimation of Word Representations in Vector Space* [online]. arXiv, 2013 [visited on 2022-05-07]. Available from DOI: 10.48550/ARXIV.1301.3781.

10. OLAH, Christopher. *Understanding LSTM Networks* [online]. 2015 [visited on 2022-05-07]. Available from: https://colah.github.io/posts/2015-08-Understanding-LSTMs/.

11. HOCHREITER, Sepp; SCHMIDHUBER, Jürgen. Long short-term memory. *Neural computation* [online]. 1997, vol. 9, no. 8, pp. 1735–1780 [visited on 2022-05-08]. Available from: http://www.bioinf.jku.at/publications/older/2604.pdf.

12. PETERS, Matthew E.; NEUMANN, Mark; IYYER, Mohit; GARDNER, Matt; CLARK, Christopher; LEE, Kenton; ZETTLEMOYER, Luke. Deep Contextualized Word Representations. In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)* [online]. New Orleans, Louisiana: Association for Computational Linguistics, 2018, pp. 2227–2237 [visited on 2022-05-08]. Available from DOI: 10.18653/v1/N18-1202.

13. ERIC, Mihail. *Deep Contextualized Word Representations with ELMo* [online]. 2018 [visited on 2022-05-08]. Available from: https://www.mihaileric.com/posts/deep-contextualized-word-representations-elmo/.

14. GALASSI, Andrea; LIPPI, Marco; TORRONI, Paolo. Attention in Natural Language Processing. *IEEE Transactions on Neural Networks and Learning Systems* [online]. 2021, vol. 32, no. 10, pp. 4291–4308 [visited on 2022-05-08]. Available from DOI: 10.1109/tnnls.2020.3019893.

15. VASWANI, Ashish; SHAZEER, Noam; PARMAR, Niki; USZKOREIT, Jakob; JONES, Llion; GOMEZ, Aidan N.; KAISER, Lukasz; POLOSUKHIN, Illia. Attention Is All You Need [online]. 2017 [visited on 2022-05-08]. Available from DOI: 10.48550/ARXIV.1706.03762.

16. VASWANI, Ashish; BENGIO, Samy; BREVDO, Eugene; CHOLLET, Francois; GOMEZ, Aidan N.; GOUWS, Stephan; JONES, Llion; KAISER, Łukasz; KALCHBRENNER, Nal; PARMAR, Niki; SEPASSI, Ryan; SHAZEER, Noam; USZKOREIT, Jakob. Tensor2Tensor for Neural Machine Translation. *CoRR* [online]. 2018, vol. abs/1803.07416 [visited on 2022-05-08]. Available from: http://arxiv.org/abs/1803.07416.

17. DEVLIN, Jacob; CHANG, Ming-Wei; LEE, Kenton; TOUTANOVA, Kristina. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding [online]. 2018 [visited on 2022-05-08]. Available from DOI: 10.48550/ARXIV.1810.04805.

18. LIU, Yinhan; OTT, Myle; GOYAL, Naman; DU, Jingfei; JOSHI, Mandar; CHEN, Danqi; LEVY, Omer; LEWIS, Mike; ZETTLEMOYER, Luke; STOYANOV, Veselin [online]. 2019 [visited on 2022-05-08]. Available from DOI: 10.48550/ARXIV.1907.11692.

19. KIM, Jin-Dong; WANG, Yue. *BioNLP Shared Tasks - Genia event extraction (GE) task* [online]. 2016 [visited on 2022-05-01]. Available from: https://bionlp.dbcls.jp/projects/bionlp-st-ge-2016/wiki/Overview.

20. WALKER Christopher, et al. *ACE 2005 Multilingual Training Corpus LDC2006T06* [online]. Philadelphia: Linguistic Data Consortium, 2006 [visited on 2022-05-01]. ISBN 1-58563-376-3. Available from DOI: 10.35111/mwxc-vh88.

21. LUAN, Yi; HE, Luheng; OSTENDORF, Mari; HAJISHIRZI, Hannaneh. Multi-Task Identification of Entities, Relations, and Coreferencefor Scientific Knowledge Graph Construction. In: *Proc. Conf. Empirical Methods Natural Language Process. (EMNLP)* [online]. 2018 [visited on 2022-05-01]. Available from: http://nlp.cs.washington.edu/sciIE/.

22. HOTTI, Alexandra; RISULEO, Riccardo Sven; MAGUREANU, Stefan; MORADI, Aref; LAGERGREN, Jens. *The Klarna Product Page Dataset: A Realistic Benchmark for Web Representation Learning* [online]. 2021 [visited on 2022-05-01]. Available from arXiv: 2111.02168 [cs.LG].

23. NI, Jianmo; LI, Jiacheng; MCAULEY, Julian. Justifying Recommendations using Distantly-Labeled Reviews and Fine-Grained Aspects. In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China: Association for Computational Linguistics, 2019, pp. 188–197. Available from DOI: 10.18653/v1/D19-1018.

24. RUOKOLAINEN, Teemu; KAUPPINEN, Pekka; SILFVERBERG, Miikka; LINDÉN, Krister. A finnish news corpus for named entity recognition. *Language Resources and Evaluation* [online]. 2019, pp. 1–26 [visited on 2022-05-01]. Available from: https://github.com/mpsilfve/finer-data.

25. RICHARDSON, Leonard. Beautiful soup documentation [online]. 2007 [visited on 2022-02-12]. Available from: https://www.crummy.com/software/BeautifulSoup/bs4/doc/.

26. *Amazon.com: Canon PIXMA MG Series All-in-One Color Inkjet Printer* [online]. 2022 [visited on 2022-01-05]. Available from: https://www.amazon.com/Inkjet-Printer-Business-Office-Resolution/dp/B09RCBFY2L/.

27. *Embeddings and Transformers – Spacy usage documentation* [online]. 2022 [visited on 2022-05-08]. Available from: https://spacy.io/usage/embeddings-transformers#embedding-layers.

28. *EntityRecognizer – Spacy usage documentation* [online]. 2022 [visited on 2022-04-22]. Available from: https://spacy.io/api/entityrecognizer.

29. *spaCy Project: Example project of creating a novel nlp component to do relation extraction from scratch.* [Online]. 2020 [visited on 2022-01-22]. Available from: https://github.com/explosion/projects/tree/v3/tutorials/rel_component.

Appendix A

Spacy configuration

```
# This is an auto-generated partial config. To use it with 'spacy train'
# you can run spacy init fill-config to auto-fill all default settings:
# python -m spacy init fill-config ./base_config.cfg ./config.cfg
[paths]
train = 'sample_data/data.spacy'
dev = 'sample_data/data.spacy'

[system]
gpu_allocator = "pytorch"

[nlp]
lang = "en"
pipeline = ["transformer","ner"]
batch_size = 128

[components]

[components.transformer]
factory = "transformer"

[components.transformer.model]
@architectures = "spacy-transformers.TransformerModel.v3"
name = "roberta-base"
tokenizer_config = {"use_fast": true}

[components.transformer.model.get_spans]
@span_getters = "spacy-transformers.strided_spans.v1"
window = 128
stride = 96

[components.ner]
factory = "ner"

[components.ner.model]
@architectures = "spacy.TransitionBasedParser.v2"
state_type = "ner"
```

```
extra_state_tokens = false
hidden_width = 64
maxout_pieces = 2
use_upper = false
nO = null

[components.ner.model.tok2vec]
@architectures = "spacy-transformers.TransformerListener.v1"
grad_factor = 1.0

[components.ner.model.tok2vec.pooling]
@layers = "reduce_mean.v1"

[corpora]

[corpora.train]
@readers = "spacy.Corpus.v1"
path = ${paths.train}
max_length = 0

[corpora.dev]
@readers = "spacy.Corpus.v1"
path = ${paths.dev}
max_length = 0

[training]
accumulate_gradient = 3
dev_corpus = "corpora.dev"
train_corpus = "corpora.train"

[training.optimizer]
@optimizers = "Adam.v1"

[training.optimizer.learn_rate]
@schedules = "warmup_linear.v1"
warmup_steps = 250
total_steps = 20000
initial_rate = 5e-5

[training.batcher]
@batchers = "spacy.batch_by_padded.v1"
discard_oversize = true
size = 2000
buffer = 256

[initialize]
vectors = ${paths.vectors}
```

A.1 Paths

The section contains paths to external files, usually development and training sets. It can contain initializations of pre-trained word vectors or pre-trained tok2vec weights. The section is re-used

as variable across the config.

```
# Version with root directory specified
[paths]
version = 5
root = "/Users/you/data"
train = "${paths.root}/train_${paths.version}.spacy"
```

These paths are usually overwritten in spacy train command:

```
python -m spacy train config.cfg --output ./output
    --paths.train ./train.spacy --paths.dev ./dev.spacy
```

A.2 System

Settings related to system and hardware. The settings are re-used across the config as variables. The most common usage is as follows:

```
[system]
seed = 0
gpu_allocator = null
```

seed – field sets the random seed used as the initialization seed for generating random numbers.
gpu_allocator – used to define library used, contains values ”pytorch”, ”tensorflow” or null.

A.3 nlp

Definition of the NLP object, its tokenizer and processing pipeline component names.
Further reading on pipelines[1].
List of all available settings for nlp[2].

```
[nlp]
lang = "en"
pipeline = ["tok2vec", "parser"]
batch_size = 1000
```

lang – contains the reference to specific language as defined in IETF languages standard[3].
pipeline – contains names of components used in pipeline, these components are further loaded in [components block]
batch_size – Default batch size to use with nlp.pipe and nlp.evaluate

A.4 nlp.tokenizer

The tokenizer is not included in the base pipeline definition, since it must be used in every NLP task. The tokenizer is used to create a Doc object with segment boundaries extracts text. We can still train our own tokenizer, but the setting has to be changed in section [nlp.tokenizer].

```
[nlp.tokenizer]
@tokenizers = "spacy.Tokenizer.v1"
```

[1] https://spacy.io/usage/processing-pipelines
[2] https://spacy.io/api/data-formats#config-nlp
[3] https://www.w3.org/International/articles/language-tags/

@tokenizers - further reading how default tokenizer works[4].

Tokenizer currently supports following languages[5].

A.5 components

It has to be defined, but it is empty in all official configurations. The interesting part happens in the nested fields of components. We have to define there all the parts of the pipeline in nested sections. Component blocks need to specify either a factory (named function to use to create component) or a source (name of path to trained pipeline to copy components from).

A.6 components.transformer

In order to use the remote pretrained models there will be allways following snippet.

```
[components.transformer]
factory = "transformer"
```

A.7 components.transformer.model

This part enables the use of transformer models in the pipeline. It supports all models that are available via the HuggingFace transformers library (with PyTorch implementation). Usually, it is needed to connect subsequent components to the shared transformer using the Transformer Listener layer.

```
[model]
@architectures = "spacy-transformers.TransformerModel.v3"
name = "roberta-base"
tokenizer_config = {"use_fast": true}
transformer_config = {}
```

- @architectures – defines the architecture to be used within the component.
- name – Model that can be loade by AutoModel[6].
- tokenizer_config – Tokenizer settings passed to AutoTokenizer[7].
- transformer_config – Transformer settings passed to AutoConfig[8].

More detailed documentation on TransformerModel.v3 [9].

A.8 components.transformer.model.get_spans

Span getters are functions that take a batch of Doc objects and return a list of Span objects for each doc to be processed by the transformer. There are 3 built-in implementations[10]: doc_spans.v1, sent_spans.v1 and strided_spans.v1.

[4] https://spacy.io/usage/linguistic-features#tokenization
[5] https://spacy.io/usage/models#languages
[6] https://huggingface.co/docs/transformers/model_doc/auto#transformers.AutoModel
[7] https://huggingface.co/docs/transformers/model_doc/auto#transformers.AutoTokenizer
[8] https://huggingface.co/transformers/model_doc/auto.html?highlight=autoconfig#transformers.AutoConfig
[9] https://spacy.io/api/architectures#TransformerModel
[10] https://spacy.io/api/transformer#span_getters

```
[transformer.model.get_spans]
@span_getters = "spacy-transformers.strided_spans.v1"
window = 128
stride = 96
```

This is configuration only for strided_spans.v1.

- window – the window size.
- streide – the stride size.

A.9 components.ner

Similarly to [components.transformer] – defaults to:

```
[components.ner]
factory = "ner"
```

A.10 components.ner.model

There is only one built-in implementation: TransitionBasedParser.v2. The model can apply to NER or dependency parsing. Transition-based parsing is an approach to structured prediction where the task of predicting the structure is mapped to a series of state transitions. The neural network state prediction model consists of either two or three subnetworks:

- tok2vec: Map each token into a vector representation. This subnetwork is run once for each batch.
- lower: Construct a feature-specific vector for each (token, feature) pair; this is run once for each batch. Constructing the state representation is simply a matter of summing the component features and applying the non-linearity.
- upper (optional): A feed-forward network that predicts scores from the state representation. If not present, the output from the lower model is used as action scores directly.

```
[components.ner.model]
@architectures = "spacy.TransitionBasedParser.v2"
state_type = "ner"
extra_state_tokens = false
hidden_width = 64
maxout_pieces = 2
use_upper = true
```

- state_type – task to extract features for. Possible values are “ner” and “parser”.
- extra_state_tokens – Whether to use an expanded feature set when extracting the state tokens. Slightly slower but sometimes improves accuracy slightly.
- hidden_width – The width of the hidden layer.
- maxout_pieces – How many pieces to use in the state prediction layer. Recommended values are 1, 2 or 3.
- use_upper – Whether to use the upper subnetwork.

More detailed documentation[11].

[11] https://spacy.io/api/architectures#TransitionBasedParser

A.11 components.ner.model.tok2vec

Subnetwork to map tokens into the vector representations.
Available built-in implementation is TransformerListener.v1[12].

```
[components.ner.model.tok2vec]
@architectures = "spacy-transformers.TransformerListener.v1"
grad_factor = 1.0
```

- grad_factor – Reweight gradients from the component before passing them upstream.

A.12 components.ner.model.tok2vec.pooling

A reduction layer is used to calculate the token vectors based on zero or more wordpiece vectors. If in doubt, mean pooling (see reduce_mean) is usually a good choice. For other options, refer to the thinc documentation on reduction operations[13].

```
[components.ner.model.tok2vec.pooling]
@layers = "reduce_mean.v1"
```

A.13 corpa

Behaves similarly to [components] but is used for definition of corpora for training, development, pretraining or other.

```
[corpora.train]
@readers = "spacy.Corpus.v1"
path = ${paths.train}
max_length = 0

[corpora.dev]

...

[corpora.pretrain]

...
```

There are two main readers, Corpus.v1 and JsonlCorpus.v1. The Corpus reader is used for reading .spacy file format, used in training and development. The JsonlCorpus is used for reading raw format, used in pretraining. For further reading, refer to Corpus readers documentation[14]. path – file path of the corpus. max_length – maximum document length, zero indicates no limit.

A.14 training

Settings and controls for the training and evaluation process.

[12] https://spacy.io/api/architectures#TransformerListener
[13] https://thinc.ai/docs/api-layers#reduction-ops
[14] https://spacy.io/api/top-level#corpus-readers

```
[training]
seed = ${system.seed}
gpu_allocator = ${system.gpu_allocator}
dropout = 0.1
accumulate_gradient = 3
# Controls early-stopping. 0 disables early stopping.
patience = 1600
# Number of epochs. 0 means unlimited. If >= 0, train corpus is loaded once in
# memory and shuffled within the training loop. -1 means stream train corpus
# rather than loading in memory with no shuffling within the training loop.
max_epochs = 0
max_steps = 20000
eval_frequency = 200
# Control how scores are printed and checkpoints are evaluated.
score_weights = {}
# Names of pipeline components that shouldn't be updated during training
frozen_components = []
# Names of pipeline components that should set annotations during training
annotating_components = []
# Location in the config where the dev corpus is defined
dev_corpus = "corpora.dev"
# Location in the config where the train corpus is defined
train_corpus = "corpora.train"
```

- seed – The random seed.
- gpu_allocator – Library for cupy to route GPU memory allocation to. It can be "pytorch" or "tensorflow".
- dropout – the dropout rate used while training NN.
- accumulate_gradient – the number of minibatch steps that are taken during training of batch.
- patience – how many update steps to take without improvement in evaluation score. 0 for disabled.
- max_epochs – maximum number of epochs.
- max_steps – maximum number of update steps for training.
- eval_frequency – how often to evaluate during training steps.
- score_weights – override the calculation metrics and their weight used to evaluate the model. Available metrics[15].
- frozen_components – list of frozen components that shouldn't be updated during training.
- annotating_components – components that should override annotations from the dataset with their own predictions during training
- dev_corpus – reference to config part with dev corpus definition.
- train_corpus – reference to config part with training corpus definition.

[15] https://spacy.io/usage/training#metrics

A.15 training.batcher

Batcher turns a stream of items into a stream of batches. After each batch, the weights of the model are updated.

```
[training.batcher]
@batchers = "spacy.batch_by_padded.v1"
discard_oversize = true
size = 2000
buffer = 256
```

There are 3 built-in batchers: batch_by_words.v1, batch_by_sequence.v1, batch_by_padded.v1. A full reference can be found in the batcher documentation[16]. Each batcher has its unique configuration. This is a configuration for batch_by_padded:

- discard_oversize – whether to discard sequences that are by themselves longer than the largest padded batch size.
- size – the largest padded size to batch sequences into.
- buffer – the number of sequences to accumulate before sorting by length.

A.16 training.optimizer

Specifies optimizer used while updating weights.

```
[training.optimizer]
@optimizers = "Adam.v1"
```

There are 3 built-in optimizers: Adam.v1, SGD.v1, RAdam.v1. More detailed documentation on optimizers[17].

A.17 training.optimizer.learn_rate

Learn rate utilizes schedules implemented in thinc library. Schedules are generators that provide different rates, schedules, decays or series. They are typically used for batch sizes or learning rates. There are seveb different implementations specified in the documentation[18].

```
[training.optimizer.learn_rate]
@schedules = "warmup_linear.v1"
warmup_steps = 250
total_steps = 20000
initial_rate = 5e-5
```

The definition of warmup linear configuration: Generate a series, starting from an initial rate, and then with a warmup period, and then a linear decline. They are used for learning rates.

A.18 initialize

Configuration block defines resources used for initialization of the pipeline. For more details, refer to the documentation[19].

[16] https://spacy.io/api/top-level#batchers
[17] https://thinc.ai/docs/api-optimizers
[18] https://thinc.ai/docs/api-schedules
[19] https://spacy.io/api/data-formats#config-initialize

Contents of the enclosed media

- `readme.txt` ..... description of media contents
- `code` ..... the directory with data, models and source codes
  - `brat` ..... the directory with brat software
  - `config` ..... the directory with prepared spaCy configs
  - `data` ..... the directory with prepared datasets
  - `model_NER` ..... the directory with trained NER model
  - `model_REL` ..... the directory with trained REL model
  - `src` ..... the directory with source codes
  - `readme.txt` ..... instructions for training and running models
  - `requirements.txt` ..... frozen python dependecies
- `thesis` ..... the directory with the source files for thesis
  - `images` ..... the directory with images used in thesis
  - `text` ..... the directory with source files for chapters in LaTeX
  - `banhetom-assignment.pdf` ..... The assignment for the first page of thesis
  - `banhetom.pdf` ..... Thesis in PDF file
  - `banhetom.tex` ..... The main LaTeX source code
  - `ctufit-thesis.cls` ..... the LaTeX template