Slovak University of Technology in Bratislava
Faculty of Informatics and Information Technologies

FIIT-XXXX-XXXXX

Bc. Mykyta Kretinin

Named Entity Recognition with Neural Networks for Text Classification

Master's Thesis

Thesis supervisor: doc. Ing. Giang Nguyen Thu, PhD.

May 2023

Slovak University of Technology in Bratislava
Faculty of Informatics and Information Technologies

FIIT-XXXX-XXXXX

Bc. Mykyta Kretinin

Named Entity Recognition with Neural Networks for Text Classification

Master's Thesis

Study programme: Intelligent Software Systems

Study field: 18. Computer Science

Training workplace:
Institute of Informatics, Information Systems and Software Engineering

Thesis supervisor: doc. Ing. Giang Nguyen Thu, PhD.

May 2023

SLOVENSKÁ TECHNICKÁ
UNIVERZITA V BRATISLAVE
FAKULTA INFORMATIKY
A INFORMAČNÝCH TECHNOLÓGIÍ

Návrh zadania diplomovej práce

Finálna verzia do diplomovej práce [1]

Študent:

Meno, priezvisko, tituly: Mykyta Kretinin, Bc.
Študijný program: Inteligentné softvérové systémy
Kontakt: xkretinin@stuba.sk

Výskumník:

Meno, priezvisko, tituly: Giang Nguyen, doc. Ing. PhD.

Projekt:

Názov:	Extrakcia entít neurónovými sieťami pre klasifikáciu textov
Názov v angličtine:	Named Entity Recognition with Neural Networks for Text Classification
Miesto vypracovania:	Ústav informatiky, informačných systémov a softvérového inžinierstva, FIIT STU, Bratislava
Oblasť problematiky:	umelá inteligencia, classification, clustering, natural language processing, named entity recognition

Text návrhu zadania[2]

V dnešnej dobe sa dennodenne vytvára veľa textových dát ako knihy, novinové alebo vedecké články. Pre extrakciu cenných informácií je dôležitá identifikácia významných pojmov z textov napr. pomocou rozpoznávania pomenovaných entít (named entity recognition, NER). Táto úloha by sa mala vykonávať ako pivotná, pretože anotované zmienky zohrávajú dôležitú úlohu pri dolovaní textu. Tradičné NER prístupy sú vo veľkej miere závislé od rozsiahlych slovníkov, cieľových pravidiel alebo dobre zostavených korpusov. Tieto metódy sú momentálne nahradené prístupom založeným na hlbokom učení, ktoré sú menej závislé od ručne vyrobených prvkov.

Analyzujte súčasný stav problematiky v oblasti spracovania a klasifikácie textu s technikami zameranými na NER s neurónovými sieťami. Navrhnite a implementujte efektívnu metódu na vytvorenie a použitie inteligentného modelu pre analýzu a klasifikáciu textových dát s použitím NER. Cieľom je vytvorenie kombinovaného prístupu NER s inými technikami spracovania prirodzeného jazyka na zlepšenie automatického procesu triedenia textových dokumentov vo vybraných štúdiách. Vyhodnoťte navrhovaný prístup a jeho výstupy pomocou dostupných metrík. Porovnajte dosiahnuté výsledky s inými existujúcimi riešeniami.

[1] Vytlačiť obojstranne na jeden list papiera

[2] 150-200 slov (1200-1700 znakov), ktoré opisujú výskumný problém v kontexte súčasného stavu vrátane motivácie a smerov riešenia

Literatúra[3]

- Li, J., Sun, A., Han, J. and Li, C., 2020. A survey on deep learning for named entity recognition. IEEE Transactions on Knowledge and Data Engineering, 34(1), pp.50-70. https://doi.org/10.1109/TKDE.2020.2981314
- KONKOL, Michal; BRYCHCÍN, Tomáš; KONOPÍK, Miloslav. Latent semantics in named entity recognition. Expert Systems with Applications, 2015, 42.7: 3470-3479. https://doi.org/10.1016/j.eswa.2014.12.015

Vyššie je uvedený návrh diplomového projektu, ktorý vypracoval(a) Bc. Mykyta Kretinin, konzultoval(a) a osvojil(a) si ho doc. Ing. Giang Nguyen, PhD. a súhlasí, že bude takýto projekt viesť v prípade, že bude pridelený tomuto študentovi.

V Bratislave dňa 26.5.2023

Podpis študenta

Podpis výskumníka

Vyjadrenie garanta predmetov Diplomový projekt I, II, III

Návrh zadania schválený: áno / nie[4]

Dňa: .....

Podpis garanta predmetov

[3] 2 vedecké zdroje, každý v samostatnej rubrike a s údajmi zodpovedajúcimi bibliografickým odkazom podľa normy STN ISO 690, ktoré sa viažu k téme zadania a preukazujú výskumnú povahu problému a jeho aktuálnosť (uveďte všetky potrebné údaje na identifikáciu zdroja, pričom uprednostnite vedecké príspevky v časopisoch a medzinárodných konferenciách)

[4] Nehodiace sa prečiarknite

I declare on my own that I have prepared this work independently, on the basis of consultations and using the mentioned literature.

In Bratislava, June 1, 2023

Bc. Mykyta Kretinin

I am grateful to my supervisor, doc. Ing. Giang Nguyen Thu, PhD., for her helpful notes, suggestions, and guidance that helped me a lot in writing this work.

Annotation

Slovak University of Technology Bratislava
FACULTY OF INFORMATICS AND INFORMATION TECHNOLOGIES
Degree Course: Intelligent Software Systems

Author: Bc. Mykyta Kretinin
Master's Thesis: Named Entity Recognition with Neural Networks for Text Classification
Supervisor: doc. Ing. Giang Nguyen Thu, PhD.
May 2023

In this thesis, the current state of named entity recognition and topic modeling methods has been observed. Different papers were mentioned to describe the history of the mentioned natural language processing methods and their use at present. Also, in this thesis were mentioned different neural network approaches related to the recognition of named entities, to show their difference, their pros and cons, and to emphasize their impact on the results. Using the described natural language processing methods, experiments were conducted to prove the positive impact of the named entity recognition models on the topic modeling methods in the context of the text classification task. For evaluation of results, a new metric has been suggested, the topic uniqueness score (TUS), derived from the cosine similarity metric. Hyperparameters of the named entity recognition models used have been tuned to achieve a good accuracy score and to obtain more quality results. The experiments conducted were done using publicly available data from the Kaggle website and news articles, for models training and testing. The results obtained have approved that the topics obtained are generally more distinct and share fewer words compared to the case of the use of a sole topic modeling method.

Keywords: natural language processing, named entity recognition, text classification, topic uniqueness

Anotácia

Slovenská technická univerzita v Bratislave
FAKULTA INFORMATIKY A INFORMAČNÝCH TECHNOLÓGIÍ
Študijný program: Intelligent Software Systems

Autor: Bc. Mykyta Kretinin
Diplomová práca: Extrakcia entít neurónovými sieťami pre klasifikáciu textov
Vedúci diplomového projektu: doc. Ing. Giang Nguyen Thu, PhD.
May 2023

V tejto práci bol sledovaný súčasný stav metód rozpoznávania pomenovaných entít a modelovania tém. Boli spomenuté rôzne práce, ktoré opisujú históriu uvedených metód spracovania prirodzeného jazyka a ich využitie v súčasnosti. Taktiež boli v tejto práci spomenuté rôzne prístupy neurónových sietí súvisiace s rozpoznávaním pomenovaných entít s cieľom poukázať na ich odlišnosť, ich výhody a nevýhody a zdôrazniť ich vplyv na výsledky. Pomocou opísaných metód spracovania prirodzeného jazyka boli vykonané experimenty, ktoré mali dokázať pozitívny vplyv modelov rozpoznávania pomenovaných entít na metódy tematického modelovania v kontexte úlohy klasifikácie textu. Na vyhodnotenie výsledkov bola navrhnutá nová metrika, skóre jedinečnosti témy (TUS), odvodené od metriky kosínusovej podobnosti. Hyperparametre použitých modelov rozpoznávania pomenovaných entít boli vyladené s cieľom dosiahnuť dobré skóre presnosti a získať kvalitnejšie výsledky. Vykonané experimenty sa uskutočnili s použitím verejne dostupných údajov z webovej stránky Kaggle a spravodajských článkov, na trénovanie a testovanie modelov. Získané výsledky potvrdili, že získané témy sú vo všeobecnosti výraznejšie a majú menej spoločných slov v porovnaní s prípadom použitia jedinej metódy tematického modelovania.

Keywords: spracovanie prirodzeného jazyka, rozpoznávanie pomenovaných entít, klasifikácia textu, jedinečnosť témy

Contents

Chapter 1

Introduction

In the digital world, there are many electronic forms of data, such as cloud storage, memory SIM cards, SSD and HDD memory, USB flash drives, etc. They help people store, retrieve, transmit, and change this information in no time and at no cost. But there is still a problem with the processing of these data, as people often do not have enough time to search for information on the Internet, compare it from different sources, or parse a large document to find particular words or sentences. For such purposes, Natural Language Processing (NLP) methods were developed, which are capable of extracting information from text data and making a brief summary, describing probable topics with their keywords, performing statistics, obtaining contextual information, etc. And, as a subfield of NLP, Named Entity Recognition (NER) methods are used to analyze a given text and retrieve words from predefined categories. These categories are, in most cases, related to the topic of the document or to the information we want to obtain from it. Therefore, for medical documents, there can be "disease", "symptom", "person", "weight", "height", "temperature", and other possibly interesting categories that may contain the required information. As these words were found and highlighted in the text, it will not only save us time reading such a document and understanding its content, but also they may be used to make and visualize statistics about people (some case studies), or to create and extend reference books about diseases and according symptoms with their rarity of occurrence, for example.

Chapter 2

Overview

2.1 Natural Language Processing

NLP is a subfield of computer science, linguistics, and sometimes Artificial Intelligence (AI), which contains many methods for the analysis and representation of human language [1]. NLP can be used for *natural language understanding*, *speech recognition*, and *natural language generation*.

2.2 Named Entity Recognition

NER is a form of NLP, which is used to identify occurrences of descriptors, mostly words, in text belonging to predefined semantic types [2, 3]. The semantic type may be a topic such as *medicine*, *sport*, or *food*. In addition, it can be a category such as *person*, *organization*, *date*, etc.

The main concept of this technique is that each text contains information about "named entities", or a real-world object, which can be denoted with a proper name. Just like real-world objects have different characteristics, which makes it possible to assign them to different groups and categories, named entities are related to particular categories or topics as well. The first time the term "named entity" was mentioned was during *MUC-6 evaluation campaign* [4], where **ENAMEX** (expressions of entity names) and **NUMEX** (numerical expression) were used for word classification. A more formal definition of named entities, as a **rigid designator** that designates the same thing in all possible worlds in which that thing exists, was given by Saul Kripke in his work "Identity and

necessity" [5].

2.2.1 Traditional approaches

Traditional approaches to NER include 3 main ways: rule-based NER, unsupervised learning, and feature-based supervised learning [6]. In the rule-based or knowledge-based approach of NER, the model classifies words based on predefined handcrafted rules, which are commonly syntactic-lexical patterns or domain-specific gazetteers. Another traditional approach to NER is unsupervised learning, where the common method used is clustering. Such a system extracts named entities from clusters of words based on context. And finally, the idea of the feature-based supervised learning approach lies in the model training process on the annotated data, from which the model learns to recognize the pattern and context of the unseen text [2].

2.2.2 Modern approach

Currently, it is very popular to do NER using Neural Network (NN), especially Deep NN. It became so popular because of its high efficiency and simplicity of usage, since such an NN is able to build a non-linear mapping between input and output, does not require very extensive domain skill and expertise to be created, and can be trained in an end-to-end paradigm. A generalized structure of a NER model based on Deep NN contains 3 broad layers [2, 6]:

1. **Distributed representations for input** (word embeddings, POS tags, etc.).
2. **Context encoder** (Convolutional Neural Network (CNN), Recurrent Neural Network (RNN), Long short-term memory (LSTM), Gated recurrent unit (GRU), transformers, ...)
3. **Tag decoder** (Softmax activation function, RNN, Conditional random field (CRF), ...)

Each of the described broad layers of the model can contain multiple "sub-layers", such that multiple LSTMs can be used in the context encoder layer, as shown in the recent survey on NER [6].

2.2.3 NER data pipeline

Before the model can be trained, the input data have to be preprocessed. This process can be split into different steps like data preparation, data preprocessing, and data loading. Similar data pipelines have been mentioned in papers [7, 8]. Sometimes the

data may be raw and not prepared to be used right away. Therefore, there is a need to modify them, transforming them into an appropriate format. There is no certain algorithm used every time, but the generalized data pipeline for a NER tasks could look similar to the one in Figure 2.1.

When the raw data are obtained, the data cleaning step will be used to remove stop words and punctuation signs, so only potentially useful words will remain. Then the sentences have to be split into separate words, since NER goal is to classify the words (tokens) in the observed text. The labeling step can be done in multiple ways, including manual labeling, classification by trained supervised/unsupervised models, transfer learning, active, or ensemble learning models. Therefore, not all data have to be labeled manually by humans if there is a model capable of doing so. However, it is important to consider the huge impact that such an automatic labeling will have on the quality of the results. It will be a big problem to train some medical nlp model on such an unreliable dataset, which may cost lives for some patients in the future. However, in most cases, labeling is not a problem of a programmer himself, as used data are commonly labeled by experts who will provide them. More information about the preprocessing steps and methods is given in the next paragraph 2.2.3

When data are cleaned and preprocessed, they should be transformed to the appropriate format. Since most RNN models expect the same input array size in the training step, all incoming arrays (sentences, paragraphs, texts) should have the same length, which can be achieved by using a padding. For example, "Endpad" words of type "Other" will be added to the end of the sentence to increase its length. Then we may consider the creation of embeddings from the input data. It is done to reduce dimensionality of the input data, which may speed up the training process without a great negative impact on the result. And, finally, dataset should be split into the training, testing, and validation, so the model will have data to train with (training), to validate its hyperparameters (validation), and to be tested and evaluated in the end (testing). The preprocessed data will be used as input to the different model steps, like training or testing.

Data preprocessing

NER models are unable to work with raw textual data of all possible content, form, and length. They need to be transformed into a numeric format to be able to understand them and process them. Additionally, raw textual data may be unstructured and contain noisy or irrelevant information. Using preprocessing techniques such an information can be removed or transformed to format, which will not negatively affect the training process

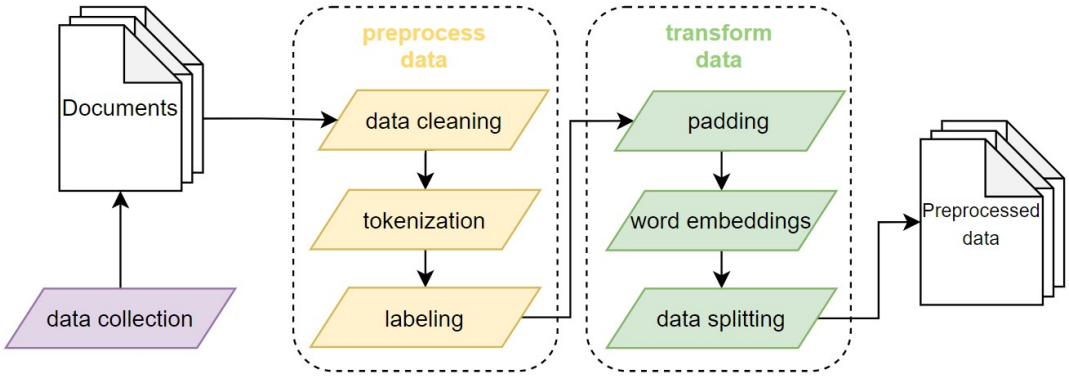

Figure 2.1: Generalized data pipeline for NER tasks

and the results. Many different methods used for text preprocessing are mentioned in the paper [9]. The most popular text preprocessing techniques include tokenization, lowercasing, and removing stop words with punctuation signs, which leads to cleaner text with tokenized words. Moreover, some preprocessing techniques may be used to reduce the dimensionality of data by removing rare and irrelevant words for the current task, or by grouping similar words together, such as word embeddings creation. It may improve the efficiency and accuracy of the model.

Data preprocessing can also contain techniques like stemming or lemmatization, which will reduce words in text to their base or root form. For example, word *'running'* will be reduced to the word *'run'*. It helps in grouping similar words together, since words like *'running'*, *'runner'* and *'run'* will be now represented by only single word - *'run'*, so the training/testing process will be faster and memory consumption will be lower. Other popular techniques of word transformation are one-hot encoding and word embeddings. The one-hot encoding (Figure 2.2) is widely used to encode categorical values as a binary vector of zeros and ones, with one being the position corresponding to the index of the category value in the vocabulary, and zeros elsewhere. Word embeddings (Figure 2.3), in contrast, represent words as dense vectors of fixed size, where similar words will have similar representations.

In summary, data preprocessing is an important step in training and testing NER models. It produces cleaner data in an appropriate format, so they can be used not only to fit input format of the model, but to accelerate training/testing speed and increase accuracy of results.

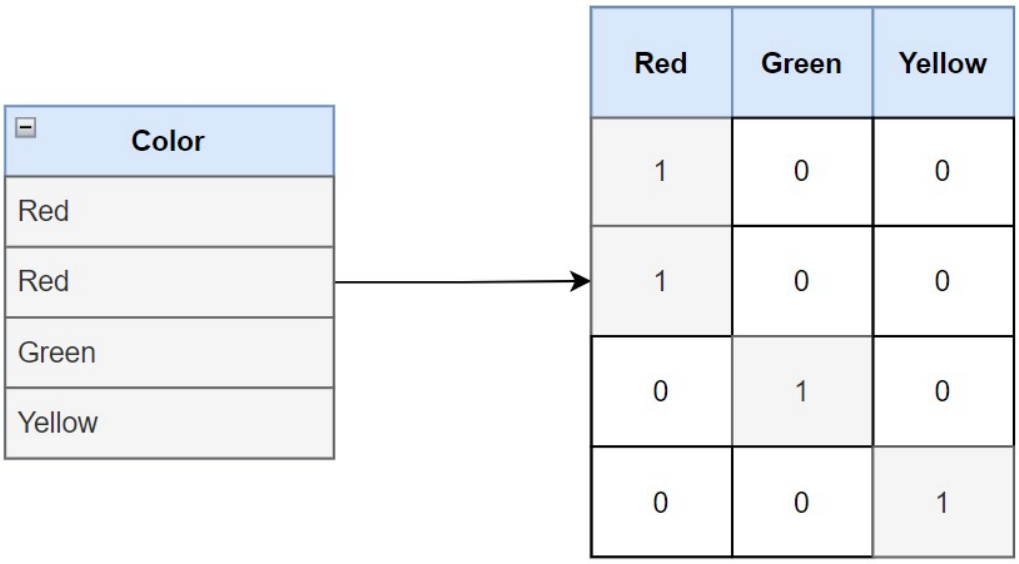

Figure 2.2: One-hot encoding example

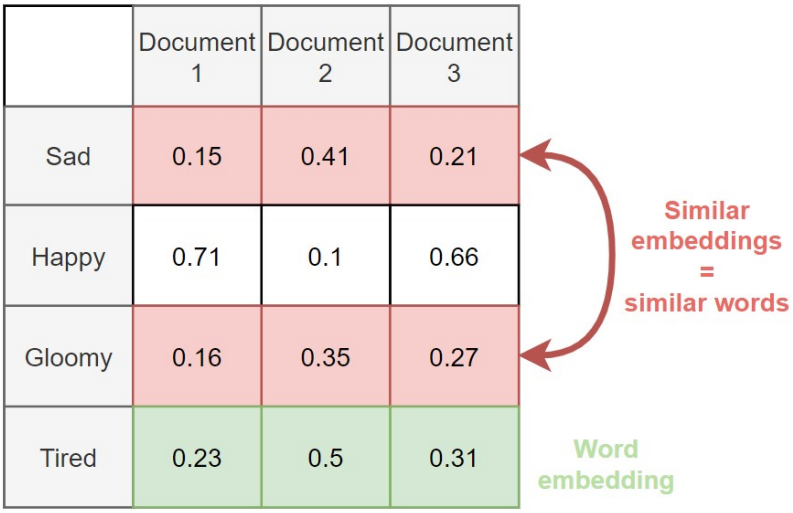

Figure 2.3: Word embeddings example

2.2.4 Deep Learning and Large Language Models (LLMs)

2.2.4.1 Recurrent Neural Network

A recurrent neural network (RNN) is a type of artificial neural network designed to handle mainly sequential and time-series data, such as natural language, speaking, or music. This concept was introduced by John Hopfield, who proposed Hopfield networks in 1982 [10].RNN models are able to work with variable-length sequences as input or output, unlike the ordinary feedforward NN architecture, which commonly has a strictly defined input and output size of a model. Because of this, RNNs become one of the most popular solutions for the tasks of NLP, machine translation, sentiment analysis, speech recognition, music generation, and many other.

RNNs consists of memory units, designed specially to handle sequential data. Each unit updates the information of the hidden state vector by remembering the current input, so that it can be used in future iterations. It allows to remember context in the analyzed data, since all related tokens, sounds or patterns will be remembered and related with each other. Therefore, the model will be able to differentiate totally identical words used in the different context. Like the weapon bow in sentence "***Bow*** *is a famous ancient weapon*" it will be separated from the action bow from the sentence "*To* ***bow*** *to someone is a sign of respect*" based on the words around it.

In addition to the hidden state vector, RNN use the same set of weights and biases for each time step and updates them during a training process. Its architecture (Figure [**f:rnn_arch**]) is very simple compared to other RNN-based architectures introduced after it. Due to its simplicity, such a model will be trained faster, compared with other similar architectures. However, there are also some limitations, and one of them is an exploding or vanishing gradient. It occurs when gradients become too large or small through a training for a long time, which makes the training process of such a model harder. Another problem of RNN is a long-time dependency - the model is prone to forget information in the hidden state vector, as it can be overwritten by the new one over time. To solve these issues, some of the RNN-based architectures were introduced later, including LSTM, GRU, bidirectional RNN, and transformers.

Long Short-Term Memory Long Short-Term Memory (LSTM) [11] is a deep learning architecture based on an RNN. It is commonly used to solve problems involving time

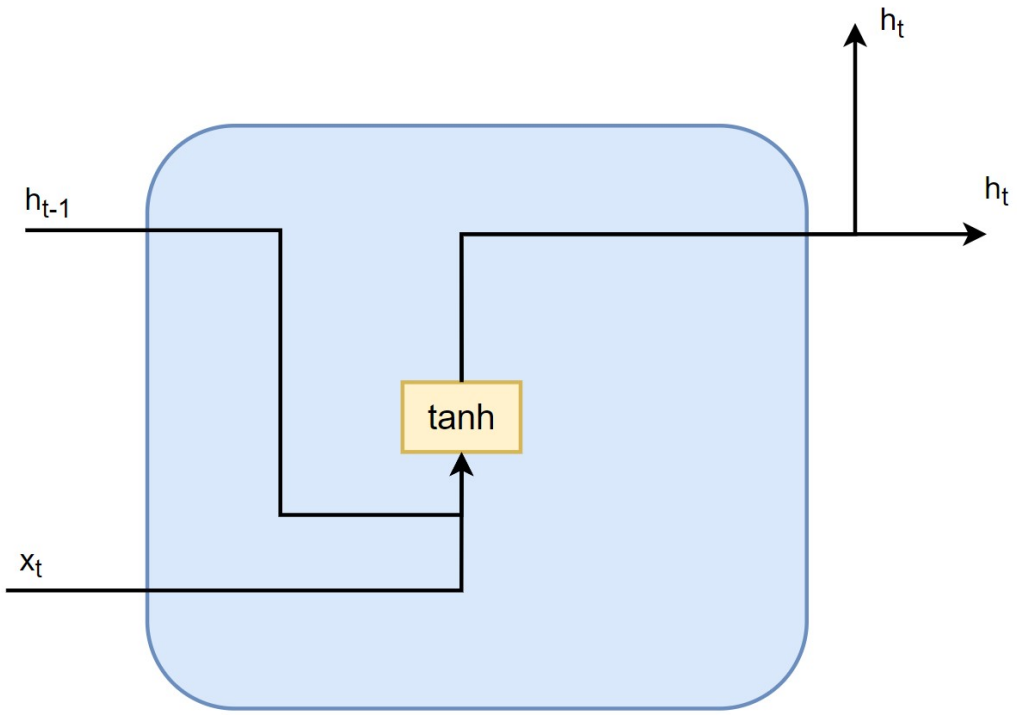

Figure 2.4: RNN memory unit structure

series and sequence data, just like RNN it was designed from. One of the reasons why it was created is to solve the vanishing gradient problem. The vanishing gradient means a situation where the gradient of the loss function becomes too small during the backpropagation, so the model cannot learn long-term dependencies. Therefore, LSTM solve this problem by adding a memory cell that stores information (word relations, context) over a long period of time and decides what to remember and what to forget when receiving new information [12].

The architecture of LSTM (Figure 2.5) consists of three gates:

1. Forget gate
2. Input gate
3. Output gate

The forget gate manages which information should be removed from memory. The input gate controls what information will be added to the memory. The output gate manages the data flow from the memory cell to the output. Each gate is implemented using a *sigmoid* activation function that returns a value in the range of 0 to 1. This value

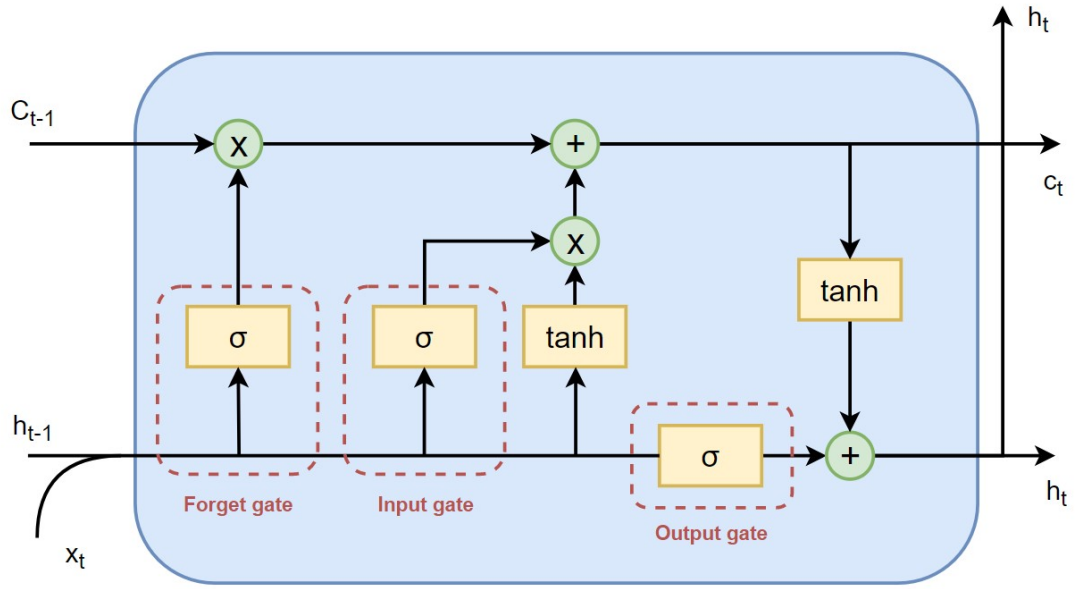

Figure 2.5: LSTM memory unit structure

is related to the amount of information which should be forgotten, added, or passed through it. As the input LSTM memory cell obtain an accumulated memory ($C_t - 1$), previous output value ($h_t - 1$) and an input value (x_t), where *t* is an index of a currently processed unit in a data sequence (eg. word).

This NN architecture is used in many different applications such as NLP, speech recognition, image capture, and video analysis. The most popular usage of LSTMs in NLP is a sentiment analysis, named entity recognition, and machine translation, since its complex architecture is able to capture more complex word relations over long time periods [13]. Also it is used in speech recognition to improve the accuracy of such a systems by capturing long-term dependencies of phonemes. In image captioning LSTM is able to generate a caption or description for the analyzed image. In video analysis, it may be used for action recognition tasks [14].

LSTM can be considered a complex model, as it performs many different calculations and has to store a lot of information about the previously processed data sequence part. This is especially true when analyzing and/or translating large text instead of small sentences. Therefore, this model may be difficult to train since it requires a lot of time and memory to be trained. Furthermore, LSTM may be prone to overfitting, since the dropout layer cannot be placed between two memory cells, but only after a whole

layer.

However, this does not mean that this model has no positive qualities. As has been said, it is able to capture and remember long-time dependencies, so the quality of machine translation goes to a whole new level. Also, this RNN modification is very flexible since it can find and store dependencies in many types of data, including textual, audio, and video [15]. Therefore, LSTM works very well for some problems and should be considered as a possible solution.

Gated Recurrent Unit A Gated Recurrent Unit (GRU) is a type of recurrent neural network (RNN) that can handle sequential data such as natural language, speech, and music. It was introduced in 2014 by Kyunghyun Cho et al. in their article [16]. As an alternative to the LSTM gated memory unit, GRU is meant to be used for similar purposes. It is hard to say which of them is generally better and which is worse, since it highly depends on the hyperparameters used, given data, and the task. It was confirmed in the article by Hitesh Hinduja [17] that LSTM produces generally better results, but its architecture is more complex and more prone to overfitting. LSTM complexity has a negative impact on the training and speed of the test of the model compared to the models using GRU. Based on the articles [18, 19], GRU is able to outperform LSTM while processing low-complexity data. Also it was stated that in the scenario of long text and small dataset GRU was 29.29% faster than LSTM, what however does not apply in other scenarios.

The architecture of GRU (Figure 2.6) consists of only two gates, which gives a lower complexity compared to LSTM:

1. Reset gate
2. Update gate

The reset gate determines how much of the remembered information has to be passed to the future. While the update gate decides how much information the memory unit will forget.

In summary, GRU can be considered a lightweight RNN alternative to LSTM, as models based on them are trained faster and give good results. However, they are not doing well in analyzing long sentences, finding more complex patterns in sequence data, and detecting long-time data relations and dependencies. For such a scenario, with long sentences and big dataset, LSTM are better to be used, as its higher complexity will be able to analyze such data better.

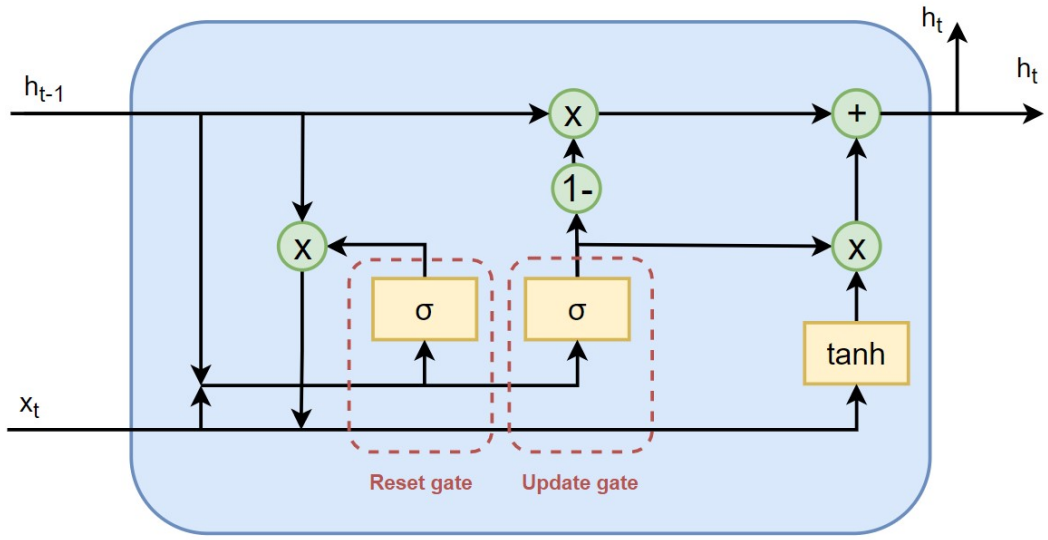

Figure 2.6: GRU memory unit structure

2.2.4.2 Transformer

Transformers are a powerful and currently very popular class of AI models that improved many different areas of machine learning, including NLP. This concept was developed by a team of Google researchers and was first introduced in their work [20] "Attention is all you need" in 2017. Different scientific papers [21] and articles [22] conducted research confirming the fact that Transformers are generally better than RNNs and are able to asses human language comprehension better than other architectures, like RNN. However, architectures like RNN have their own advantage over Transformers, which is a more efficient inference due to the constant computational and memory complexity. Therefore, authors of the work [23] "RWKV: Reinventing RNNs for the Transformer Era" decided to develop a new architecture named Receptance Weighted Key Value (RWKV) that combines the efficient inference of RNN architecture with parallelization of Transformers. Like that, the created model of such an architecture can be formulated as either a Transformer or an RNN, as it includes both their advantages, and managed to perform on par with similarly sized Transformers.

The key idea of transformers lies in their ability to capture relations between different sequence elements by attending to all other elements at the same time. Unlike RNN, where sequences are processed step-by-step, Transformers use the attention mechanism to assign weight to every element in a sequence. Like that, it is able to indicate a relative

importance of the element, to be used later in the prediction or text generation tasks based on the other elements. Therefore, with this attention mechanism, Transformers are able to capture and model long-range dependencies and understand more complex patterns in the observed data.

Transformers are widely used in different applications in various areas of AI. As mentioned above, they generally outperformed RNN architecture, and significantly improved the areas of sentiment analysis, language translation, named entity recognition, question-answer system, and many other problems of NLP. One of the most famous examples of the Transformer-based model is Bidirectional Encoder Representations from Transformers, which achieved state-of-the-art performance in many different language understanding tasks. Also, Transformers are widely used in various image processing tasks [24], where they also achieved good performance. By converting input images into sequence data, they can process visual and multidimensional information on the same principles as they process text. This approach obtained good results in different tasks, including image capture, object detection, and image classification [25]. Moreover, transformers are also being used in recommendation systems and have managed to obtain good results in this area [26]. Since Transformer is able to capture complex patterns in data and long-time dependencies between objects, such a system, when trained on large enough datasets, will be able to make accurate predictions for new users and items. In addition to that, transformers are used in audio processing tasks [27], since speech and music are related to sequential data and are perfect for such an architecture. There is a good example of the combination of the recommendation system with the audio processing by transformers described in the paper [28]. Here the authors used BERT for a bidirectional training of user's sequence to make recommendations. It is trained on the encoder part of the Transformer model, where the attention mechanism learns contextual relations between past interactions of the user.

Additionally, it is worth mentioning that Transformers can be adapted to be used on different tasks or data through a process called transfer learning, where the pre-trained Transformer model is being trained on smaller, task-specific related datasets. Using such an approach, we improve generalization of the model and reduce the needed amount of the labeled data and computational resources, which is beneficial for Transformers, since they are highly dependent on the input data and have relatively high memory complexity. Currently many researchers and companies use pre-trained generalized models, like GPT-3 or GPT-4, and fine-tune them to the selected specific tasks, such as summarization or creative writing. Perfect examples of a currently popular application

are ChatGPT developed by OpenAI and based on the GPT-3.5 and GPT-4 versions, and Bing: Chat based on the GPT-4 developed by Microsoft. These applications became popular due to their ability to interact with the user and answer the given question in the required format. Therefore, such a generalized transformer-based model includes many different subtasks, including text generation, language translation, sentiment analysis, text summarization, and question answering.

In summary, the best versatility of transformers lies in the ability to adapt and fine-tune to many different specific tasks. Today they are widely used in many areas of AI and currently achieved results mentioned in different articles and scientific papers were good enough to consider the replacement of old architectures like RNN or CNN by a fine-tuned Transformer model.

2.2.4.3 BERT

Bidirectional Encoder Representations from Transformers is a state-of-the-art model in the area of NLP, that has revolutionized this field since its introduction by the Google research team in their work [29] "Bert: Pre-training of deep bidirectional transformers for language understanding" in 2018. BERT is a deep learning model which is based on a transformer architecture and is able to capture contextual relations between elements in text more effectively than previous NLP models, which has been described in the previous subsection about Transformers.

One of the key differences between BERT and a typical Transformer lies in the tasks for which they are designed. Original transformers model [20] were designed mainly for sequence-to-sequence tasks, such as machine translation, where the input sequence and output may have different lengths and are being processed in parallel. It is done by using an encoder-decoder architecture with an attention mechanism, which maps words from the input sequence to a vector of a fixed length and then decoder generates output from such a vector. BERT, on the other hand, is not designed to generate data from input sequences and is used to create high-quality representations of the input text that later can be used for a variety of NLP tasks and with different architectures. Since BERT produces contextualized word embedding, where each representation vector values are based on the context of the word in the sentences it appeared, they can be used by the Transformer decoder to generate a summarization for an input text, for example. Of course, BERT's usage is not limited by Transformer, as it is used for other architectures, including RNN, LSTM and GRU, and for tasks such as text summarization, named entity recognition, or sentiment analysis.

Another difference between BERT and Transformers lies in the training process. Classic Transformer models are trained on supervised learning tasks, where data are labeled, and the input and output sequences are aligned. On the contrary, BERT is being trained on unsupervised tasks, where the model learns to predict missing words in the text sequence, or a sentence in the documents [30].

2.3 Applications of NER

NER methods have been used for a long time, and some of the models created even achieved near-human performance. For example, the best system entering **MUC-7** receives 93.39% of F-score metric, while human annotators obtained 97.60% and 96.95% [31].

Since its creation, this approach of text analysis has been widely used for different purposes, such as information extraction, question-answering, machine translation or text summarization [32], and many more. In addition, it is very useful in biomedical data processing, as it helps retrieve biomedical entities related to text content, such as genes, chemicals, diseases [33], or patient data from Electronic Health Record (EHR). Some other works aim to adjust existing NLP approaches to work with NER [34], or develop models for work with languages other than English [35].

Therefore, it can be said that the NER methods have great potential, as they are used as a "basement" for many text processing approaches [32], and can support many different languages. Therefore, any improvement and research done in this field may have a positive impact on existing methods, resulting in greater effectiveness, more complex language structure support, or simply higher accuracy of the results.

2.4 Unsupervised Learning and Text Classification with NER

Unsupervised learning is a type of Machine Learning (ML) learning where model is trained on data that are not labeled. Therefore, the model must find patterns and relations in the data by its own, without getting any certain labels to compare results with, and without any help from the human side as well.

There are several possible ways to classify text, which includes Latent Dirichlet Allocation (LDA) and NER methods application. LDA is an unsupervised ML method by itself, and it is one of the most well known and popular unsupervised methods for text classification, among its predecessors (Latent Semantic Indexing (LSI), Non-negative

Matrix Factorization (NMF)) and successors (Correlated Topic Model (CTM)). It shows great scalability, low time complexity and good results, which has been confirmed in some articles [36, 37].

However, there is no guaranteed context in topics created by LDA, since words in them are chosen only by a statistical calculation of their co-occurrences. Therefore, it may be a good solution to try and use NER methods as well, so the relations between words can be found out and will be added to the process of topics creation. As a result, topics will have more words related by context, which should make them more realistic-like instead of purely statistical. This approach is experimented with in this paper, to decide whether the combination of NER approach with text classification methods such as LDA will improve the quality of the result topics.

2.5 Metrics for a topic modeling evaluation

2.5.1 Perplexity

Perplexity is a common metric used in ML to evaluate language models [38], either statistical ones like LDA and NMF, or models based on a neural network such as RNN and transformers. This metric measures how well a probability distribution of the model can predict a given sample [39]. Intuitively, as its name states, we can describe it as a measurement of how the 'perplexed' model is seeing a given token or text. Therefore, the low perplexity value indicates that our probability distribution of the model is good at predicting the given text. In contrast, in the case of high perplexity, the model may lack data to predict a given sample well enough.

$$perplexity = 2^H \tag{2.1}$$

where

H is an entropy 2.2 of the language model;

This metric can be computed as a 2 raised to the power of entropy (H, Equation 2.2 (from [39])), which is an average number of bits we need to represent each word or token from the predicted sequence. Higher entropy values indicate a less accurate model, which will lead to higher perplexity as well. Therefore, our goal is to lower the perplexity value by lowering the model's entropy.

$$H = \sum_{b_n} p(b_n) log_2 p(w_n|b_{n-1}) \tag{2.2}$$

where

H is an entropy of the language model;
b_n is a block of continuous letters in the text;
w_n is an **n**th letter, right after the b_n block;

In summary, perplexity is one of the most common metrics used to evaluate language models. It is calculated from the entropy of the model, and the less value of it indicates that the used model is more accurate and is less 'perplexed' with a given text or word. Also, if perplexity is low, it can indicate that model is good enough to predict the next letter of the word or a word in the sentence, giving us a way to evaluate text-generating models as well.

2.5.2 Coherence

Coherence is a quantitative metric that is used to evaluate the logical and semantic coherence in a part of the text. It evaluates how well the terms and information are connected and organized in the text. This metric is designed to represent how meaningful an analyzed text is for the reader and how easily he can follow and understand the written ideas.

There are many different ways to measure the coherence score of the text. One of them is based on the concept of topic continuity, where it is examined how consistent and coherent discussion on a specific topics, pieces of text, are in the analyzed documents. This refers to the use of smaller parts of the text, cohesive devices at the phrase or word level, to create connections between larger parts, sentences and paragraphs [40]. Therefore, the lexical cohesion can be achieved by repetition of the appropriate words, key terms, pronouns, synonyms, and antonyms, where they have to be. And this metric calculates the appropriateness of these coherence devices, evaluating the continuity and interconnection of the discussion topics of the text.

Another possible way to use coherence is to use it as a loss function in the model training process [41]. Since the most common way to evaluate the text classification model is to do it calculating the perplexity score, it may not reflect the measure of coherence of text. Therefore, when we train a model that classifies or generates textual data, coherence

can be calculated on the pieces of observed or generated text, and this score will be used to improve the training process.

From the description above, it can be concluded that coherence has a wide usage in several applications, text classification, machine translation, and text generation. Its common usage is a text coherence evaluation, which let researchers and developers to compare created models, therefore creating more quality products. Moreover, since coherence can be applied to the parts of text, it may help not only evaluate a whole model but also find local problems with observed data, thus helping to discover smaller and harder to find problems with the model.

2.5.3 Topic Uniqueness Score

To check whether NER is capable of improving the quality and precision of the text classification results, several experiments were conducted and evaluated. Text classification in this case means not a sentiment analysis or any other example of the binary/multilabel classification, but a topic assignment to the given text. Since topics are quite dynamic and versatile, as they can change over a lifetime of the model, it is hard to use supervised learning. Therefore, chosen topic modeling models have to be trained on unlabeled data.

Since our main goal is to measure the impact of NER on topic modeling, we should choose a suitable metric for it that will evaluate exactly the affected model part. In this case, we were expecting NER to add a context relation between the words in the dictionary and the topics produced by the chosen topic modeling model. Since we are talking about context, classical metrics for models like LDA, - perplexity and coherence -, are out of question as they are unable to catch a context between words in the topics. Therefore, we suggest using a custom variation of a *cosine similarity* metric, named Topic Uniqueness Score (equation 5.1). This metric is described in more detail in Section 5.1.

2.6 Related Work

Each existing method in NLP has its own pros and cons. Some of them depend greatly on a data format: time-sequence or dimensional data, distribution of the data, missing values, and many other similar factors. Therefore, different studies on the efficiency of NLP methods were conducted. For example, Treviso et al. (2018) in their work "Efficient Methods for Natural Language Processing: A Survey" [42] conducted a survey on the

currently used NLP methods and a general data pipeline to describe how efficient they can be when used with limited resources. Also, there is article [43] from Gutierrez Daniel (2021), where he described different NLP methods, including transformers, and how they are efficiently used at certain data for different tasks. And that is one of the reasons why it may be beneficial in some cases to combine different NLP approaches, as they are able to achieve even a better result.

One of the examples of combined approaches was described in the work "LDA in Character-LSTM-CRF Named Entity Recognition" [44], where the authors presented a way to improve a NER task by combining character sequence encoding and decoding models, which use the LSTM layer as the base, with the LDA topic modeling method to improve NER system accuracy of results. The authors believe that the topic models of LDA will improve the ability of the NER system to capture context in long-range documents.

Another example of a combination of NLP methods was mentioned in the work "Recognizing named entities in agricultural documents using LDA-based topic modeling techniques" [45]. There, the authors created an Agriculture Named Entity Recognition system using Topic Modeling (AERTM) with agricultural vocabulary (AGROVOC) to identify agriculture-related terminology in the text, such as crops, soil types, pathogenes, crop diseases and fertilizers. However, the designed NER model was unable to identify words related to soil types, crop diseases, and fertilizer categories. Therefore, they decided to use LDA to identify these entities in the text. Their LDA and NER combined model managed to achieve 80% accuracy on the data obtained from reputed agricultural websites, which confirmed the efficiency of the added LDA method.

2.7 Summary

. . .

2.8 Research Starting Points

The purpose of this work is to investigate the positive impact of NER (RNN, LSTM, GRU) on text classification methods like LDA, NMF or Formal Concept Analysis (FCA).

The research value of this work lies in the comparison of the results obtained using combinations of NER and different text classification approaches with a classic model.

Such an output will help future work oriented to text classification tasks, where the context of the words may be important.

The implemented solutions will be compared between themselves and with the classical text classification methods. Obtained results and models' efficiency will be evaluated using suitable metrics.

Chapter 3

Objectives and methodology

3.1 Objectives

The main objective of this work is to find a possible way to integrate a classic NER approach into topic modeling methods to improve the quality of text classification results. The suggested approach should be tested, and its results must be evaluated. To approve the efficiency of the suggested method, the results should be compared with those of the classical approach.

Therefore, this work aims to achieve the following objectives:

- Observe the impact of the NER on the chosen topic modeling method
- Test the suggested NER integration into topic modeling with different NER model architectures
- Find advantages and limitation of the suggested approach

3.2 Methodology

To achieve objectives of the work, the following steps should be taken:

- Find data to be used in the experiments
- Train a few NER models with different architectures (Long short-term memory, Gated recurrent unit, transformers, ...)
- Create a topic modeling ML model
- Select a suitable metric(s) to evaluate results' quality
- Compare results achieved by chosen topic model with the same one using NER

- Explain pros and cons of the integration of NER into topic modeling methods

Chapter 4

Design

...

Chapter 5

Implementation

5.1 Topic Uniqueness Score metric

The most popular metrics for evaluating topics produced by topic modeling methods are perplexity and coherence. But they are purely statistical and only display a co-occurrence of words in the produced topics, which do not represent if those words are really related somehow or are just happen to be in the same documents. For this reason, in this work we suggest using a Topic Uniqueness Score metric, which is based on the cosine similarity distance between word probabilities distributions in the produced topics.

$$TUS = \frac{1}{N} \sum_{n=1}^{N} cosine\_sim(P_{topic}, P_{other}) \tag{5.1}$$

where

N is a number of topics in the model;
P_{topic} is a word probability distribution for a topic;
P_{other} are words probability distributions for topics other than topic;

The suggested metric will find a similarity of word distributions overall over the topics of the observed model. Intuitively we can say, if two topics (word probabilities distributions) are similar, then they will have higher score - closer to 1. Our goal is to create more distinct topics, as they will have less intersection area (shared words) and,

therefore, contain more related words. If TUS score of the combined method (NER + topic modeling) will be lower than that of a classic approach (topic modeling only), the experiment will be successful.

Chapter 6

Experiments and Evaluation

NER is widely used to identify named entities in text, so classifying them as certain predefined categories. On the other hand, there are methods to classify a whole text, so that it will be able to understand its general topic in mere seconds. Recent studies [42, 43, 44, 45] show that each NLP method has its own pros and cons;and that it is possible to combine different NLP methods so that a greater result can be achieved or the accuracy of the result will be enhanced. One of such approaches is an idea to use LDA to make better word embeddings, so the words within will be better connected, as their connections will be based on the documents analyzed. And these embeddings can be used to increase quality of a NER model, as it will be given a better 'starting point', where there are already some connections between words, so there is a pseudo-context between them.

In this work, it is suggested to use a trained NER model to enhance LDA model results, so a purely statistical method will get a bit of context understanding, which suppose to lead to a more coherent and understandable for humans topics. Therefore, to test the efficiency of this approach, a NER model based on a deep NN was created, which was trained to detect words of certain topics from annotated training datasets. Then this NER model has been used to preprocess the textual data used for training an LDA model. The core idea is to detect words from certain categories and then manually increase their weights by creating separate documents containing only these words and adding these artificial documents to the original training dataset. In this way, words from the same category will be grouped, and they will have a higher chance to appear on the same topic of the LDA model.

6.1 NER model architecture

At first, a new NER model has to be created. Its architecture (Figure 6.1) consists of the Input, Embedding, Bidirectional LSTM, Dropout and Dense layers.

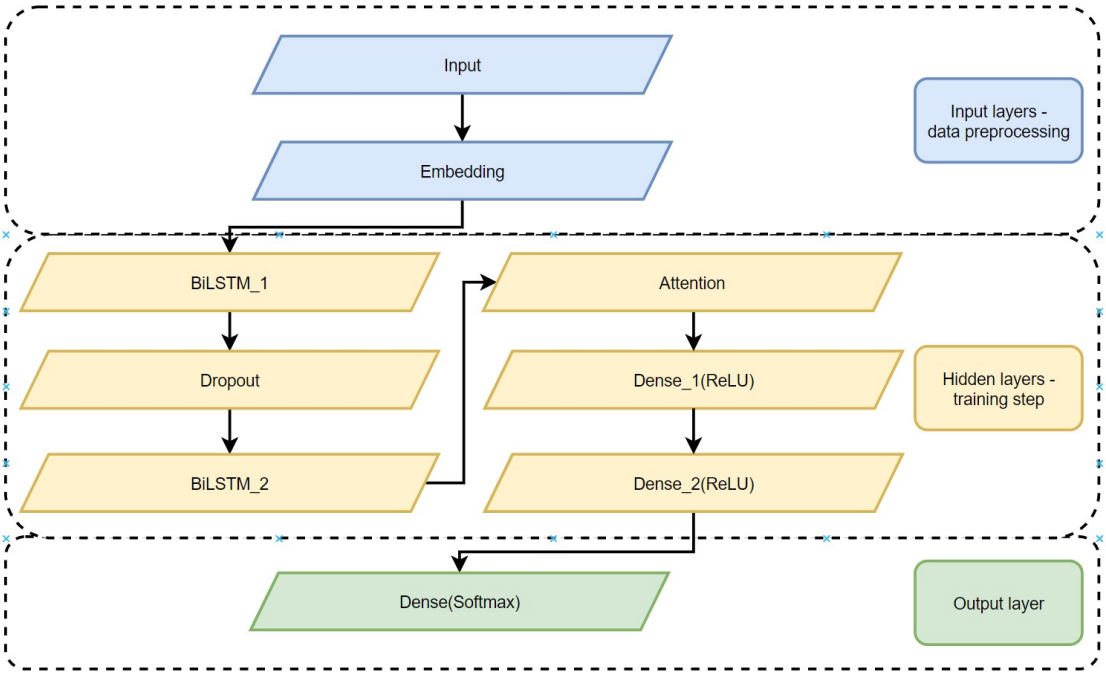

Figure 6.1: NER model's deep NN architecture

The input layer is used to specify the expected input vector dimensions, while the Embedding layer is used to transform the input data before the training process. After the data have been obtained and processed, the model will use them to train and create contextual connections between different words. That is what 2 BiLSTM layers are used for. To generalize a model and to reduce chances of overfitting, the Dense layer is used after the first BiLSTm layer, so some part of the discovered relations between words will be erased. And in the end there are 2 Dense layers, which meant to discover latent features between found word relation, so the final model will be able to analyze words and their combinations more clearly. And at the last Dense layer, the model will decide the analyzed word category probabilities, so it is possible to extract the most probable one. Such a structure will have enough complexity to catch hidden patterns and word relations in the analyzed sentences. It is supposed to create a good generalized model that can be used not only with the training dataset, but also with other unseen data as well.

1. Input Layer
2. Embedding Layer
3. Bidirectional LSTM Layer
4. Dropout Layer
5. Bidirectional LSTM Layer
6. Attention layer
7. Dense Layer
8. Dense Layer
9. Dense(output, softmax) Layer

6.2 Used data

In the experiments conducted data were used from open sources. The first experiment was conducted using the Kaggle dataset[1] for the training of the NER model and Reuter's articles on medical topic for the LDA model creation.

6.2.1 GDPR

All external data were used for study and scientific non-commercial purposes only. Kaggle datasets were not published as part of this work and can be accessed using the links mentioned in this section. Reuters permits to use their data for personal non-commercial purposes and does not permit their data being published.

6.3 Experiments and Evaluation

6.3.1 NER impact on LDA topics

Main idea

The field of NLP advanced quite a lot in recent years. Different techniques and methods have been developed to work with the textual data, extract different information from there, and create a new text non-recognizable from the human writings. And one of such areas of development was NER, where named entities are extracted based on the context of the text analyzed. Modern NER approaches mostly include the usage of deep NN or transformers, as they give more accurate results than some of the older methods, such as gazetteers and rule-based systems.

[1] https://www.kaggle.com/datasets/sushilkumarinfo/cord19processeddataset

However, it is not enough to identify these entities in the text, because sometimes it is crucial to get their relations in the context of the whole document or even a set of documents. And one of the possible ways to achieve this is to use the LDA model, one of the most popular unsupervised techniques of text classification. It is used to identify latent topics in the analyzed document by grouping words within by their co-occurrence. But one of the biggest problems of LDA and other similar models is a lack of context. Because most of them are not able to understand relation between words, since they look only at words occurrence statistics.

One of the possible ways to increase LDA model efficiency and add 'context' to those latent topics is to use the NER model. Using it, it is possible to find named entities related with each other by category and context and artificially create new documents with words grouped by a category they belong to. This way, after newly created documents will be added to the analyzed dataset, words co-occurrence will be artificially manipulated to contain relations between named entities, and so will be final topics. It will lead to more distinct and unique topics than those that a classic LDA model will be able to produce. For an easier understanding of the object relations in the suggested method, you can see the diagram 6.2.

NER model

To confirm this theory NER model was trained on a medical CORD-19 annotated dataset from Kaggle[2]. It contains different annotated words categorized by one of 64 different labels. NER model's architecture was the same as that described in Section 6.1 (6.1)

After the hyperparameter tuning step, the best model managed to achieve 91.8% accuracy on a validation dataset, which is not a very high score, but should be enough to confirm or reject the usefulness of the suggested approach. Therefore, this trained model has been used in the experiments.

6.3.1.1 LDA vs. NER & LDA

To confirm whether NER is actually able to improve topic uniqueness and quality, experiments were carried out to compare the results of a classic LDA approach, with a combination of NER and LDA.

To evaluate results, TUS (equation 5.1), a custom variation of the cosine similarity metric, has been used. As can be seen from the Figure 6.4, TUS value achieved with a suggested

[2]https://www.kaggle.com/datasets/sushilkumarinfo/cord19processeddataset

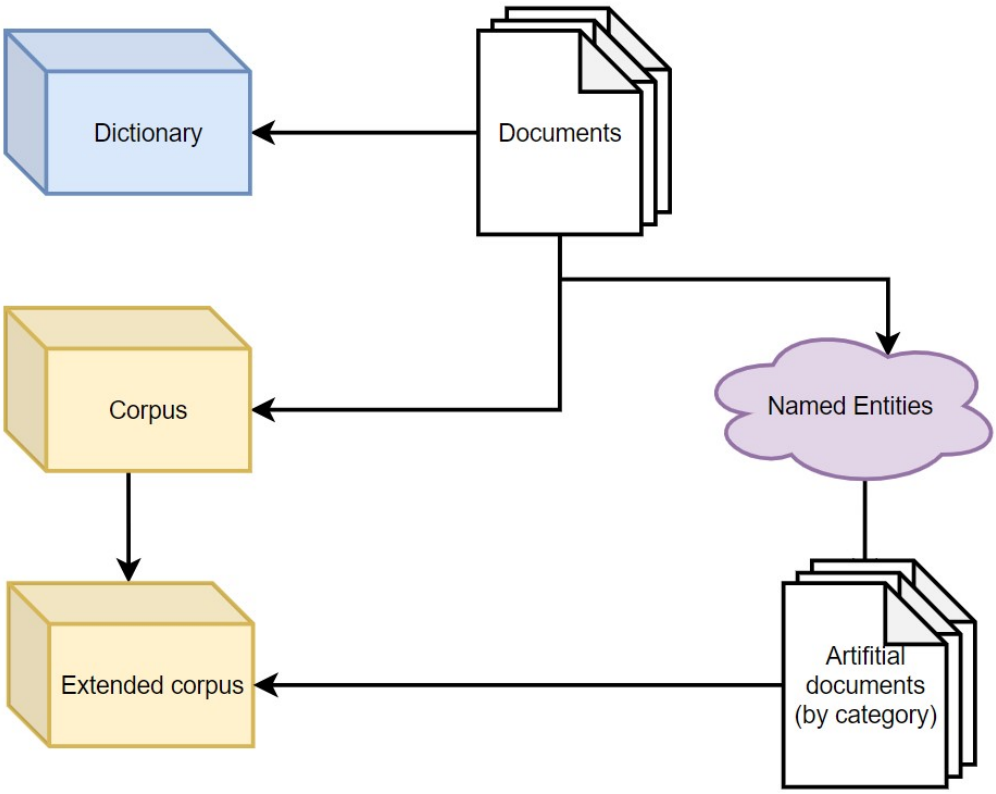

Figure 6.2: Diagram representation of a suggested solution

approach of the combination of NER and LDA combination was better than a classic one, LDA only. The combined approach managed to obtain lower TUS value for each configuration, with a different number of topics. Therefore, we can state that using NER with topic modeling methods like LDA is able to improve topic uniqueness by artificially manipulating word distributions in documents, increasing the chance that contextually related words will be in the same topic.

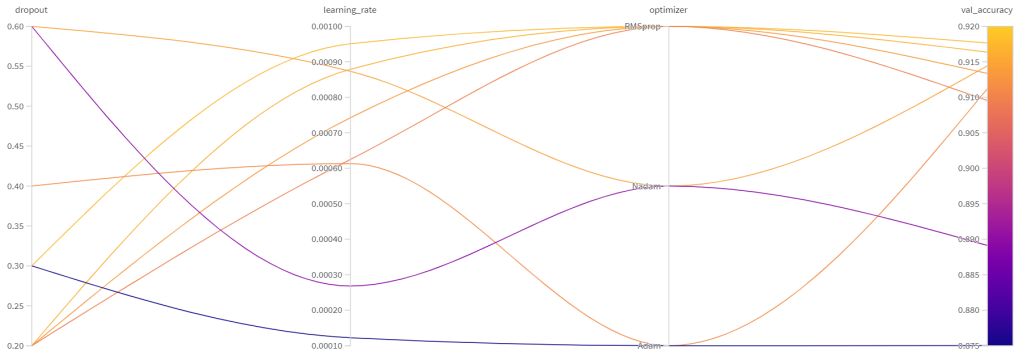

Figure 6.3: NER tuning results

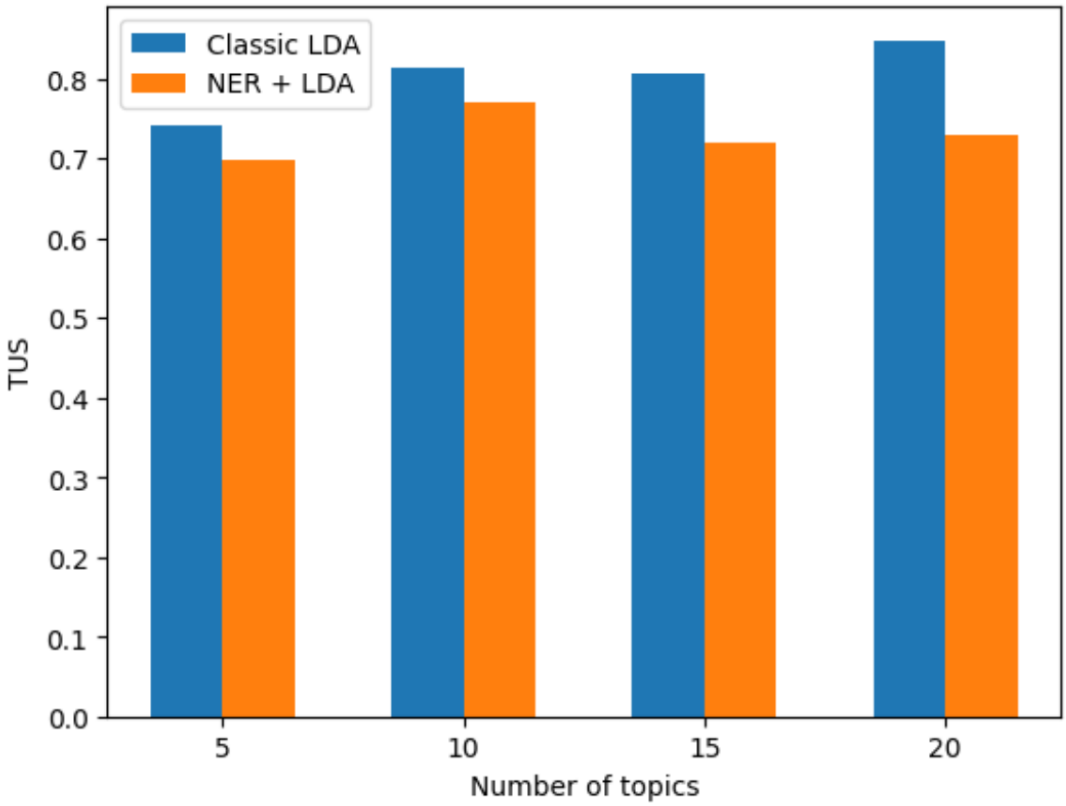

Figure 6.4: LDA vs. NER & LDA, TUS value

6.4 Limitations

The suggested method should be used only with NER models, where categories (word labels, classes) can be named as a topic. For example, if model is able to detect entities of the DATE and PERSON categories, then create separate documents with them and integrate into the topic modeling model may not be a good idea, as is the most cases such entities as well-known dates or people are a part of some broader topic, like 'sports', 'medicine', 'politics' and many others. Therefore, it will make more sense to create topics from entities, which categories can be called as a topic of some documents and articles.

6.5 Summary

Chapter 7

Conclusion

7.1 Summary

...

7.2 Future Work

...

Resumé

slovak

. . .

List of Abbreviations

AI Artificial Intelligence

BERT Bidirectional Encoder Representations from Transformers

CNN Convolutional Neural Network

CRF Conditional random field

CTM Correlated Topic Model

EHR Electronic Health Record

FCA Formal Concept Analysis

GRU Gated recurrent unit

LDA Latent Dirichlet Allocation

LSI Latent Semantic Indexing

LSTM Long short-term memory

ML Machine Learning

NER Named Entity Recognition

NLP Natural Language Processing

NMF Non-negative Matrix Factorization

NN Neural Network

RNN Recurrent Neural Network

TUS Topic Uniqueness Score

List of Figures

List of Tables

References

1. CAMBRIA, Erik; WHITE, Bebo. Jumping NLP curves: A review of natural language processing research. *IEEE Computational intelligence magazine*. 2014, vol. 9, no. 2, pp. 48–57. Available from DOI: 10.1109/MCI.2014.2307227.
2. LI, Jing; SUN, Aixin; HAN, Jianglei; LI, Chenliang. A Survey on Deep Learning for Named Entity Recognition. *IEEE Transactions on Knowledge and Data Engineering*. 2022, vol. 34, no. 1, pp. 50–70. Available from DOI: 10.1109/TKDE.2020.2981314.
3. MARSHALL, Christopher. *What is named entity recognition (NER) and how can I use it?* DATAVERSITY, 2019. Available also from: https://medium.com/mysuperai/what-is-named-entity-recognition-ner-and-how-can-i-use-it-2b68cf6f545d.
4. GRISHMAN, Ralph; SUNDHEIM, Beth. DESIGN OF THE MUC-6 EVALUATION. 1996. Available also from: https://aclanthology.org/M95-1001.pdf.
5. KRIPKE, Saul A. Identity and Necessity. In: MUNITZ, Milton Karl (ed.). *Identity and Individuation*. New York: New York University Press, 1971.
6. YADAV, Vikas; BETHARD, Steven. A survey on recent advances in named entity recognition from deep learning models. *arXiv preprint arXiv:1910.11470*. 2019.
7. CAHUANTZI, Roberto; CHEN, Xinye; GÜTTEL, Stefan. A comparison of LSTM and GRU networks for learning symbolic sequences. *arXiv preprint arXiv:2107.02248*. 2021.
8. SHERSTINSKY, Alex. Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network. *Physica D: Nonlinear Phenomena*. 2020, vol. 404, p. 132306.
9. TABASSUM, Ayisha; PATIL, Rajendra R. A survey on text pre-processing & feature extraction techniques in natural language processing. *International Research Journal of Engineering and Technology (IRJET)*. 2020, vol. 7, no. 06, pp. 4864–4867.
10. HOPFIELD, John J. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*. 1982, vol. 79, no. 8, pp. 2554–2558.

11. HOCHREITER, Sepp; SCHMIDHUBER, Jürgen. Long short-term memory. *Neural computation*. 1997, vol. 9, no. 8, pp. 1735–1780.
12. STAUDEMEYER, Ralf C; MORRIS, Eric Rothstein. Understanding LSTM–a tutorial into long short-term memory recurrent neural networks. *arXiv preprint arXiv:1909.09586*. 2019.
13. YAO, Lirong; GUAN, Yazhuo. An Improved LSTM Structure for Natural Language Processing. In: *2018 IEEE International Conference of Safety Produce Informatization (IICSPI)*. 2018, pp. 565–569. Available from DOI: 10.1109/IICSPI.2018.8690387.
14. *Long Short-Term Memory (LSTM)* [Saturn Cloud]. [N.d.]. Available also from: https://saturncloud.io/glossary/lstm/.
15. SHASHIDHAR, R; KULKARNI, Sudarshan Patil. Integration of Audio video Speech Recognition using LSTM and Feed Forward Convolutional Neural Network. 2021.
16. CHO, Kyunghyun; VAN MERRIËNBOER, Bart; GULCEHRE, Caglar; BAHDANAU, Dzmitry; BOUGARES, Fethi; SCHWENK, Holger; BENGIO, Yoshua. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*. 2014.
17. HINDUJA, Hitesh. *From Pre-RNNs to GPT-4: How Large Language Models are Changing NLP*. 2023. Available also from: https://pub.towardsai.net/from-pre-rnns-to-gpt-4-how-large-language-models-are-changing-nlp-43512ce7e765.
18. CAHUANTZI, Roberto; CHEN, Xinye; GÜTTEL, Stefan. A comparison of LSTM and GRU networks for learning symbolic sequences. *arXiv preprint arXiv:2107.02248*. 2021.
19. YANG, Shudong; YU, Xueying; ZHOU, Ying. LSTM and GRU Neural Network Performance Comparison Study: Taking Yelp Review Dataset as an Example. 2020, pp. 98–101. Available from DOI: 10.1109/IWECAI50956.2020.00027.
20. VASWANI, Ashish; SHAZEER, Noam; PARMAR, Niki; USZKOREIT, Jakob; JONES, Llion; GOMEZ, Aidan N; KAISER, Łukasz; POLOSUKHIN, Illia. Attention is all you need. *Advances in neural information processing systems*. 2017, vol. 30.
21. MICHAELOV, James A; BARDOLPH, Megan D; COULSON, Seana; BERGEN, Benjamin K. Different kinds of cognitive plausibility: why are transformers better than RNNs at predicting N400 amplitude? *arXiv preprint arXiv:2107.09648*. 2021.
22. SRIVASTAVA, Niharika. *How Transformers work in deep learning and NLP: an intuitive introduction?* 2023. Available also from: https://www.e2enetworks.com/blog/how-transformers-work-in-deep-learning-and-nlp-an-intuitive-introduction.

23. PENG, Bo; ALCAIDE, Eric; ANTHONY, Quentin; ALBALAK, Alon; ARCADINHO, Samuel; CAO, Huanqi; CHENG, Xin; CHUNG, Michael; GRELLA, Matteo; GV, Kranthi Kiran, et al. RWKV: Reinventing RNNs for the Transformer Era. *arXiv preprint arXiv:2305.13048*. 2023.
24. KIM, Boah; KIM, Jeongsol; YE, Jong Chul. Task-Agnostic Vision Transformer for Distributed Learning of Image Processing. *IEEE Transactions on Image Processing*. 2023, vol. 32, pp. 203–218. Available from DOI: 10.1109/TIP.2022.3226892.
25. BOESCH, Gaudenz. *Vision Transformers (ViT) in Image Recognition*. 2023. Available also from: https://viso.ai/deep-learning/vision-transformer-vit.
26. ZANG, Runqiang; ZUO, Meiyun; ZHOU, Jilei; XUE, Yining; HUANG, Keman. Sequence and Distance Aware Transformer for Recommendation Systems. In: *2021 IEEE International Conference on Web Services (ICWS)*. 2021, pp. 117–124. Available from DOI: 10.1109/ICWS53863.2021.00027.
27. LATIF, Siddique; ZAIDI, Aun; CUAYAHUITL, Heriberto; SHAMSHAD, Fahad; SHOUKAT, Moazzam; QADIR, Junaid. Transformers in speech processing: A survey. *arXiv preprint arXiv:2303.11607*. 2023.
28. YADAV, Naina; SINGH, Anil Kumar. Bi-directional encoder representation of transformer model for sequential music recommender system. In: *Forum for Information Retrieval Evaluation*. 2020, pp. 49–53.
29. DEVLIN, Jacob; CHANG, Ming-Wei; LEE, Kenton; TOUTANOVA, Kristina. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*. 2018.
30. DELUCIA, Sally-Ann. *Unleashing the Power of BERT: How the Transformer Model Revolutionized NLP*. 2023. Available also from: https://arize.com/blog-course/unleashing-bert-transformer-model-nlp/.
31. MARSH, Elaine; PERZANOWSKI, Dennis. *MUC-7 EVALUATION OF IE TECHNOLOGY: Overview of Results*. 1998. Available also from: https://www-nlpir.nist.gov/related_projects/muc/proceedings/muc_7_proceedings/marsh_slides.pdf.
32. GOYAL, Archana; GUPTA, Vishal; KUMAR, Manish. Recent Named Entity Recognition and Classification techniques: A systematic review. *Computer Science Review*. 2018, vol. 29, pp. 21–43. ISSN 1574-0137. Available from DOI: https://doi.org/10.1016/j.cosrev.2018.06.001.
33. HABIBI, Maryam; WEBER, Leon; NEVES, Mariana; WIEGANDT, David Luis; LESER, Ulf. Deep learning with word embeddings improves biomedical named

entity recognition. *Bioinformatics*. 2017, vol. 33, no. 14, pp. i37–i48. ISSN 1367-4803. Available from DOI: 10.1093/bioinformatics/btx228.

34. YAN, Hang; DENG, Bocao; LI, Xiaonan; QIU, Xipeng. TENER: adapting transformer encoder for named entity recognition. *arXiv preprint arXiv:1911.04474*. 2019.
35. SOUZA, Fábio; NOGUEIRA, Rodrigo; LOTUFO, Roberto. Portuguese named entity recognition using BERT-CRF. *arXiv preprint arXiv:1909.10649*. 2019.
36. ETAATI, Farhood. *Text Classification using LDA*. 2020. Available also from: https://medium.com/analytics-vidhya/text-classification-using-lda-35d5b98d4f05.
37. KIM, Sang-Woon; GIL, Joon-Min. Research paper classification systems based on TF-IDF and LDA schemes. *Human-centric Computing and Information Sciences*. 2019, vol. 9, pp. 1–21.
38. *Perplexity of fixed-length models*. [N.d.]. Available also from: https://huggingface.co/docs/transformers/perplexity.
39. HUYEN, Chip. *Evaluation Metrics for Language Modeling*. 2019. Available also from: https://thegradient.pub/understanding-evaluation-metrics-for-language-models/.
40. ALYOUSEF, Hesham Suleiman. An SF-MDA of the textual and the logical cohesive devices in a postgraduate accounting course. *SAGE Open*. 2020, vol. 10, no. 3, p. 2158244020947129. Available from DOI: 10.1177/2158244020947129.
41. KRASNASHCHOK, Katsiaryna; CHERIF, Aymen. Coherence regularization for neural topic models. In: *Advances in Neural Networks–ISNN 2019: 16th International Symposium on Neural Networks, ISNN 2019, Moscow, Russia, July 10–12, 2019, Proceedings, Part I 16*. Springer, 2019, pp. 426–433.
42. TREVISO, Marcos; JI, Tianchu; LEE, Ji-Ung; AKEN, Betty van; CAO, Qingqing; CIOSICI, Manuel R; HASSID, Michael; HEAFIELD, Kenneth; HOOKER, Sara; MARTINS, Pedro H, et al. Efficient methods for natural language processing: a survey. *arXiv preprint arXiv:2209.00099*. 2022.
43. GUTIERREZ, Daniel. *Top Recent NLP Research* [ODSC]. 2021. Available also from: https://opendatascience.com/top-recent-nlp-research/.
44. KONOPÍK, Miloslav; PRAŽÁK, Ondřej. LDA in character-LSTM-CRF named entity recognition. In: *Text, Speech, and Dialogue: 21st International Conference, TSD 2018, Brno, Czech Republic, September 11-14, 2018, Proceedings 21*. Springer, 2018, pp. 58–66.
45. GANGADHARAN, Veena; GUPTA, Deepa. Recognizing named entities in agriculture documents using LDA based topic modelling techniques. *Procedia Computer Science*. 2020, vol. 171, pp. 1337–1345.

Appendix A

Description of Digital Submission

Thesis Evidence Number:
FIIT-XXXX-XXXXX

Name of submitted archive:
`DP_MykytaKretinin.zip`

The digital part included in the thesis contains the following files: ...

Appendix B

Technical Documentation

B.1 Installation Manual

B.2 User Manual

Appendix C

Work Schedule

C.1 Work Schedule for Master's Thesis 1

1^{st}-2^{nd} week	Topic consultation. Study on Named Entity Recognition techniques and recent works.
3^{rd}-4^{th} week	Find data for the first experiment. Train and evaluate NER model.
5^{th}-6^{th} week	Consultations. Case study on a possible solutions of the chosen problematic. Choose metrics to use in the experiments.
7^{th}-9^{th} week	Finish the experiment. Presentation of intermediate results.
10^{th} week	Work on the document report. Start writing state-of-the-art (SOTA) part. Document the conducted experiment.
11^{th}-12^{th} week	Consultations. Finishing the documentation report for Master's thesis 1 (DP1). Creating the working plan for Master's thesis 2 (DP2).

Master's thesis 1 work plan realization is considered as satisfied and successful. During the first weeks of the semester, I became acquainted with the state of the art of the master thesis. During that time, relevant articles were found to be used later as references, a base program code was written, and suitable data were found to be used later in the experiments. After that, during the fifth and sixth weeks, a case study was conducted a case study on possible solutions for a chosen problem, and suitable metrics were found which can be used in the experiments. When the first experiment was finished, I began to work on the state-of-the-art (SOTA) part, which has been

mostly finished until the end of semester. During the last weeks, a work review was conducted to find grammar mistakes and add formal content. The planned work plan has been successfully completed to the end of the semester. However, there is still room for improvement in the future semesters, including conducting more experiments and describing the design and implementation of the proposed approach.

C.2 Work Schedule for Master's Thesis 2

1st week	Revision of the work plan for the DP2. Revision state-of-the-art part part of the work, add missing information.
2rd-4th week	Conduct the next experiment. Try to find more metrics for evaluation of results. Consultations.
5th-6th week	Describe obtained results from the last experiment. Add charts and diagrams for better understanding of the results.
7th-9th week	Finish Design and Implementation sections. Consult the achieved results.
10th-12th week	Finishing the documentation report for Master's thesis 2 (DP2). Consultations. Creating the working plan for Master's thesis 3 (DP3).

Master's thesis 2 work plan realization is considered as satisfied and successful. To be added . . .

C.3 Work Schedule for Master's Thesis 3

1st week	Revision of the work plan for the DP3. Research of the selected approaches. Consultations. Containerization of experiments
2nd week	Presenting the updates of the selected frameworks. Further research on state-of-the-art privacy preserving techniques. Initial design for extending existing case studies on . Containerization of experiments.
3rd-4th week	Presentation of intermediate results. Consultations.
5th-6th week	Development and implementation of the changes discussed. Consultations. Presentation of final results. Documentation of archived results.
7th-9th week	Consultations. Finalizing the implementation of case studies. Work on the thesis documentation report.
10th-12th week	Finalizing of the thesis documentation report. Consultations.

Master's thesis 3 work plan realization is considered as satisfied and successful. To be added . . .