UNIVERSITATEA NAȚIONALĂ DE ȘTIINȚĂ ȘI TEHNOLOGIE POLITEHNICA BUCUREȘTI

FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE

DEPARTAMENTUL CALCULATOARE

|  |  |
| --- | --- |
| upb | cs |

PROIECT DE DIPLOMĂ

Titlul proiectului de diplomă (ex: Șablon proiect de diplomă)

Subtitlu (ex: versiunea 2018)

Ioana Popescu

**Coordonator științific:**

Prof. dr. ing. Andrei Ionescu

BUCUREŞTI

2018

UNIVERSITY POLITEHNICA OF BUCHAREST

FACULTY OF AUTOMATIC CONTROL AND COMPUTERS

COMPUTER SCIENCE DEPARTMENT

|  |  |
| --- | --- |
| upb | cs |

DIPLOMA PROJECT

Diploma Project Title (eg: Diploma project template)

Subtitle (eg: 2018 version)

Ioana Popescu

**Thesis advisor:**

Prof. dr. ing. Andrei Ionescu

BUCHAREST

2018

**CUPRINS**

[Sinopsis 2](#_Toc168927500)

[Abstract 2](#_Toc168927501)

[Mulțumiri 3](#_Toc168927502)

[1 Introducere 4](#_Toc168927503)

[1.1 Context 4](#_Toc168927504)

[1.2 Problema 4](#_Toc168927505)

[1.3 Obiective 4](#_Toc168927506)

[1.4 Structura lucrării 4](#_Toc168927507)

[2 Analiza și specificarea cerințelor 5](#_Toc168927508)

[3 STATE OF THE ART 6](#_Toc168927509)

[3.1 YAKE! (Yet Another Keyword Extractor) 6](#_Toc168927510)

[3.2 Latent Dirichlet Allocation 9](#_Toc168927511)

[3.3 spaCy 12](#_Toc168927512)

[3.3.1 The spaCy pipeline 12](#_Toc168927513)

[4 Soluția propusă 13](#_Toc168927514)

[4.1 Indicații formatare formule 14](#_Toc168927515)

[5 Detalii de implementare 15](#_Toc168927516)

[5.1 Indicații formatare tabele 15](#_Toc168927517)

[6 Studiu de caz / Evaluarea rezultatelor 17](#_Toc168927518)

[7 Concluzii 19](#_Toc168927519)

[8 REFERENCES 20](#_Toc168927520)

[9 Anexe 21](#_Toc168927521)

# Sinopsis

Sinopsisul proiectului are rol de introducere, conținând atât o descriere pe scurt a problemei abordate cât și o enumerare sumară a rezultatelor și a concluziilor. Se recomandă ca sinopsisul să fie redactat într-un limbaj accesibil unei persoane nefamiliarizate cu domeniul, dar în același timp destul de specific pentru a oferi rapid o vedere de ansamblu asupra proiectului prezentat.

Sinopsisul proiectului va fi redactat atât în română cât și în engleză. Ca dimensiunea recomandată aceasta secțiune va avea maxim 200 de cuvinte pentru fiecare variantă. Împreună, ambele variante se vor încadra într-o singură pagină.

# Abstract

The abstract has an introductory role and should engulf both a brief description of the issue at hand, as well as an overview of the obtained results and conclusions. The abstract should be formulated such that even somebody that is unfamiliar with the projects’ domain can grasp the objectives of the thesis while, at the same time, retaining a specificity level offering a bird’s eye view of the project.

The projects’ abstract will be elaborated in both Romanian and English. The recommended size for this section is limited to 200 words for each version. Together, both versions will fit in one page.

# Mulțumiri

(opțional) Aici puteți introduce o secțiunea specială de mulțumiri / acknowledgments.

# Introducere

Parametrii de formatare recomandați pentru lucrare:

* Font recomandat: Calibri; Dimensiune font: 12;
* Spațiere între linii: 1,15; Spațiere după paragraf: 8pt;
* Stil: Justified;
* Dimensiune pagină: A4; Margini: 2,54cm/ 2,54cm/ 2,54cm/ 2,54cm;
* Heading1: Calibri, 14, bold, all caps;
* Heading2: Calibri, 14, bold;
* Heading3: Calibri, 12.
* Font pentru formule: Cambria Math, 12.

În cadrul introducerii, este necesară abordarea următoarelor puncte care reprezintă de fapt familiarizarea cititorului (comisia, alți colegi sau experți în domeniu) cu tema proiectului, soluția propusa și cuprinsul/structura lucrării. Deși introducerea poate conține și unele elemente mai generale, se recomandă păstrarea unui limbaj tehnic, specific audienței care va citi lucrarea.

În cadrul capitolelor următoare, veți regăsi o serie notații de forma [Dezvoltare de produs], [Cercetare]. Acest tip de formatare este utilizat exclusiv în acest template pentru a marca sfaturi și cerințe specifice pentru lucrări de diploma cu specific diferit. În pregătirea documentului vostru, nu veți utiliza aceste marcaje.

Elementele pe care trebuie să le abordați în introducere sunt descrise în cadrul subcapitolelor de mai jos.

## Context

O scurtă introducere a proiectului, motivație, explicație de ce este relevant domeniul proiectului.

## Problema

Care este problema pe care proiectul o va rezolva.

## Obiective

Care sunt obiectivele proiectului/soluției/abordării/ideii; Ce creșteri sau evoluții determină rezolvarea proiectului.

## Structura lucrării

Un paragraf în care fiecare dintre secțiunile următoare este prezentată în 1-2 fraze, punând accentul pe elementele cele mai semnificative din fiecare secțiune.

# Analiza și specificarea cerințelor

[Dezvoltare de produs] Acest capitol va analiza cerințele produsului din prisma potențialilor clienți și a scenariilor de utilizare preconizate, urmând a fi generată o lista de funcționalități.

[Cercetare] Acest capitol va introduce motivația realizării proiectului propus.

Dacă proiectul de licență face parte dintr-un proiect mai amplu (de exemplu un proiect complex, la care lucrează 2 studenți (ex: 1 student la front-end-ul aplicației, 1 student la back-end-ul aplicației), în acest capitol va fi explicat pe scurt ansamblul proiectului și ce parte din proiect este adresată de lucrarea propusă.

Criterii pentru calificativul Nesatisfăcător:

* [Dezvoltare de produs] Cerințele sunt imaginate de student pe baza unei analize a pieței;
* [Cercetare] Nu se oferă o motivație valida.

Criterii pentru calificativul *Satisfăcător*:

* [Dezvoltare de produs] Există un interviu, un client, analiza cerințelor este elaborată pe baza interviului;
* [Cercetare] Motivația este doar personala.

Criterii pentru calificativul *Bine*:

* [Dezvoltare de produs] Proces iterativ pe baza unor interviuri cu mai mulți clienți, dezvoltare MVP, reevaluare cerințe;
* [Cercetare] Motivația este legata de o necesitate științifica / tehnica explicită.

# STATE OF THE ART

In the field of Natural Language Processing (NLP), keyword extraction is essential for text mining purposes, such as automatic indexing, automatic summarization, automatic classification, automatic clustering, automatic filtering and topic detection (Zhang, 2008). The necessity of accurately extracting keywords and topics from documents has led to the development of numerous algorithms. However, this area is still not as precise as other computer science fields (Campos et al., 2018). There is no universal approach for effectively extracting keywords and topics without requiring either large manually labelled data sets for training or a comprehensive preprocessing of the text, according to the specifics of the domain and of the individual document. Therefore, one needs to use more than one tool in order to achieve accurate results. In this chapter, I will discuss the tools integrated in my project: YAKE! (Yet Another Keyword Extractor) for keyword extraction, LDA (Latent Dirichlet Allocation) for topic modelling, the spaCy Python library for preprocessing, part of speech tagging and named entity recognition, fastText for language detection and the Gensim package for the LDA implementation.

## YAKE! (Yet Another Keyword Extractor)

Supervised methods have been the prevailing approach in keyword extraction, but they require extensive training and manual work for annotating large collections of documents (Campos et al., 2020). Moreover, these machine learning models may not perform well on texts from domains that it was not trained on.

Despite being an unsupervised method, TFIDF (Term Frequency-Inverse Document Frequency) also needs an extended collection of documents since it assumes that a term is relevant in a specific text if it occurs frequently in that document but is otherwise rare in all of the documents. The formula used to measure the significance of a word is (Nomoto, 2022):

(1)

The first term in equation (1) is the Term Frequency (frequency of word in document j), while the second term is known as the Inverse Document Frequency, where the numerator represents the total count of documents and the denominator is the number of documents that contain word (Nomoto, 2022).

RAKE (Rapid Automatic Keyword Extraction) is another unsupervised keyword extractor that makes use of a co-occurrence matrix, thus rewarding with a higher score the words that appear not only frequently, but also in longer phrases (Rose et al., 2010):

(2)

In formula (2), the degree (deg) is the total length of the potential keyphrases that word w appears in, and the frequency represents the count of w in those sentences.

The score of a candidate keyphrase is computed by adding up the scores of the component words.

The subsequent unsupervised algorithm is YAKE, the approach that I chose to pursue in my implementation. YAKE ranks candidate keywords using a score calculated based on statistical text features. It does not rely on a pre-labelled corpus and it is equally effective regardless of domain or language, only requiring a stopwords list specific to the language it is applied on (Campos et al., 2020). YAKE’s ability of operating on individual documents without the necessity of additional corpora makes it suitable for extracting keywords from a set of technical abstracts, a continuously expanding and changing collection (Rose et al., 2010).

O imagine care conține text, captură de ecran, Font, document

Descriere generată automat

Figure 1: YAKE Algorithm (Campos et al., 2020)

Figure 1 presents YAKE as pseudocode, highlighting the five steps of the algorithm (Campos et al., 2020):

1. Text pre-processing and candidate term identification

The text is split into sentences, sentences are split into chunks delimited by punctuation marks and chunks are divided into tokens. Then, stopwords are removed and tokens are tagged with the following labels: d (digit or number), u (unparsable), a (acronyms), u (uppercase), p (parsable).

1. Feature extraction

The algorithm favours tokens that appear repeatedly in uppercase or abbreviated as an acronym and assumes a term is more important if it can be found closer to the beginning of the text. Therefore, it proceeds by calculating some statistics for each term: the frequency, the number of times that the word occurs capitalized inside sentences, the number of times that the term appears as an acronym and the positions of the sentences that contain it in the document.

Afterwards, YAKE calculates five features for each token and combines these attributes into the following formula which quantifies the significance of a token:

(3)

In formula (3), evaluates the casing dimension, taking into account the frequency of occurrence starting with a capital letter and the frequency of the acronym. The factor measures the importance of a word based on all of its positions in the text, more precisely the indices of the sentences it occurs in. The “term frequency normalization” () represents the frequency of a term, normalized to ensure that, in large documents, this element does not overweigh the other features in the formula. The fourth component, “term relatedness to context” (), aims to downgrade the words that have similar characteristics to those of stopwords, specifically a high number of occurrences with many different words around them. The last feature in the formula is (“term different sentence”), which rewards tokens that appear in numerous sentences.

1. Computing term score

The score of each token is determined using formula (3), with the most significant terms having the lowest score.

1. Generating n-grams and calculating the scores of potential keyphrases

In the fourth stage, the YAKE algorithm finds potential keywords by extracting n-grams (with the maximum n given as input to the algorithm) from the chunks obtained in the first step, choosing only the phrases that do not contain tokens marked as “digit” or “unparsable” and do not start or end with a stopword.

The score of a keyword kw is determined as:

(4)

In equation (4), the multiplied scores of the component tokens of kw are divided by the sum of these scores amplified by the frequency KF of the keyword. Consequently, the algorithm can differentiate between potential keywords that contain the same words but reordered and choose the most frequent one.

1. Handling duplicates

After ranking the keywords by the lowest score, YAKE removes duplicates using a similarity measure, which can be either the Levenshtein similarity, the Jaro-Winkler similarity or the sequence matcher. Two keywords are considered duplicates if their distance, computed using one of the three previous methods, surpasses a deduplication threshold chosen before starting the algorithm and given as parameter. When two keyphrases exceed this similarity threshold, the one with the higher score is removed from the final keyword list.

## Latent Dirichlet Allocation

Topic modeling methods aim to discover “hidden” (latent) topics from large collections of documents. Each topic consists of a set of words that frequently occur together and thus it is assumed that they are conceptually related. Given its capacity of capturing semantic features of corpora, topic modeling is an effective tool in text mining and information retrieval, used in text summarization and sentiment analysis applications (Abdelrazek et al., 2023).

There are four categories of topic models: algebraic, fuzzy, probabilistic and neural (Abdelrazek et al., 2023).

The most notable algebraic topic modeling method is Latent Semantic Indexing (LSI) (Deerwester et al., 1990), that represent a collection of documents as a term-document matrix, where an element of the matrix is the frequency of a word in a document (a row corresponds to a word, while documents are placed on columns). Deerwester et al. (1990) decomposed the term-document matrix using singular value decomposition (SVD) in order to obtain a vectorial representation for words and texts. Geometrically, the distance (calculated as cosine similarity) between these vectors indicates the degree of semantic correlation between two terms, two documents or a term and a document. Although LSI is proficient in capturing synonymy, it does not handle polysemy with the same efficacy (Deerwester et al., 1990).

LSI is not based on a solid statistical foundation and a flaw that it presents is the false assumption that words and documents follow a joint Gaussian distribution (Abdelrazek et al., 2023). These concerns are solved by the probabilistic approaches, which, according to Abdelrazek et al. (2023), dominated the field before neural topic models began their rising around 2015.

Latent Dirichlet Allocation (LDA), a generative probabilistic model developed by Blei et al. (2003), remains one of the most popular topic modeling methods. LDA is characterized by a series of assumptions:

* documents and words are exchangeable, their order does not matter, concept known as the bag of words model (Blei et al., 2003);
* each document is a probability distribution over topics and the distributions of topics in all documents share the same Dirichlet prior (Jelodar et al., 2017);
* each topic is a distribution over words and the distributions of words in all the topics are characterized by a common Dirichlet prior (Jelodar et al., 2017).

The generative process of LDA begins by choosing the Dirichlet priors α and β, corpus-level parameters. Afterwards, for each document, the algorithm chooses the distribution θ of topics in the current document from a Dirichlet distribution of parameter α. Then, for each of the N words of the document, a topic is sampled from distribution θ and a word is chosen from the distribution of this topic (Blei et al., 2003). Thus, the generative process of LDA iteratively calculates the probability distribution of topics in documents and the distribution of words within topics, by randomly assigning them based on a probability distribution, until the results converge.

Griffiths & Steyvers (2004) used LDA to extract topics from scientific abstracts and achieved meaningful results, as in the following figure:

O imagine care conține text, Font, alb, chitanță

Descriere generată automat

Figure 2: An abstract with the words labelled according to the topic they belong in

In figure 2, the words of an abstract are tagged with a number corresponding to the topic they belong in. The highlighted words are extracted from the topic with the highest probability; hence they can be considered as keywords of the abstract and, according to Griffiths & Steyvers (2004), these words accurately summarize the content of the document.

The limitations of LDA stem from the variability of its parameters which need to be selected before running the algorithm: the Dirichlet priors α and β and the number of topics. Although these parameters can be fine-tuned using Gibbs Sampling, their variations significantly impact the performance of the algorithm (Griffiths & Steyvers, 2004). A higher value for α leads to a larger number of topics being assigned to a document, while the value of β controls the sparsity of the topic-word distributions (a higher β determines the algorithm to assign similar probabilities to more words). Regarding the number of topics, a small number may lead to underfitting, while a large number of topics generally results in overfitting the data (Tijare & Rani, 2020).

Another considerable issue is that LDA requires a thorough clean-up and preprocessing of the documents; stopword removal and lemmatization are very important, otherwise stopwords would occur among the words extracted from topics and different forms of the same word would be considered separately (Moreno-Ortiz, 2024).

Tijare & Rani (2020) compared the results of LDA on social media data, using the implementations from popular Python packages Gensim and scikit-learn, and visualizing them with pyLDAvis, a tool which makes graphical representations of the topics and computes a coherence score, a measure of the accuracy of the solution.

O imagine care conține linie, Interval, diagramă, captură de ecran

Descriere generată automat

Figure 3: Coherence score for scikit-learn LDA (Tijare & Rani, 2020)

Figure 3 shows the coherence score depending on the number of topics. Despite the rising, Tijare & Rani (2020) conclude that, above 10 topics, they begin to overlap.

O imagine care conține captură de ecran, cerc, diagramă, proiectare

Descriere generată automat

Figure 4: pyLDAvis representation of LDA results obtained with Gensim

In figure 4, which depicts 10 topics extracted using the Gensim implementation of LDA, topics 7, 9 and 10 slightly intersect, indicating they share common terms.

Tijare & Rani (2020) also compared LDA with LSI and NMF, another topic modeling technique, and came to the conclusion that LDA achieves better coherence scores. They used two methods for computing a coherence score, with U\_MASS resulting in a coherence of for LDA and for LSI (negative values suggest coherence; the lower the value, the more coherent the topics) and C\_V showing a score of for LDA and for LSI (LDA has the higher score, thus the higher coherence).

## spaCy

spaCy is a state-of-the-art natural language processing Python library, that offers a customizable processing pipeline, with components capable of tokenization, lemmatization, recognizing named entities, tagging parts of speech and dependency parsing (*SpaCy*, n.d.):

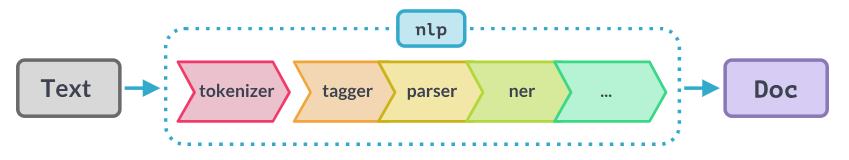


Figure 5: spaCy pipeline

Figure 5 shows the default components of the spaCy pipeline, which can be modified by disabling some components or by adding custom components developed by the user or available in the open-source community.

The motivation behind spaCy is explained by its creator, Matthew Honnibal, who wanted to develop a fast, efficient and compact language processing tool, suitable for production environments, as opposed to the older libraries, which he believes offer too many features that are otherwise nor effective nor up to date with the advancements in computer science (Matthew Honnibal, n.d.). Honnibal explained why he did not contribute to popular NLP library NLTK instead of starting a new project, stating that he thinks the maintainers of NLTK should not keep adding to the project, but “throw almost all of it away”.

The reason why numerous NLP packages, NLTK included, became so large yet not industry-grade is simply because this is not their target, they are developed by researchers for researchers and students (Srinivasa-Desikan, 2018).

In the following sections, I will detail the default components of the spaCy pipeline, specifically the components of the trained model that I used in my project, “en\_core\_web\_lg”, a large general-purpose pipeline, trained on web articles, with 514,000 word vectors (*SpaCy*, n.d.).

O imagine care conține text, captură de ecran, diagramă, linie

Descriere generată automat

Figure 6: en\_core\_web\_lg pipeline (*SpaCy*, n.d.)

The components of the common spaCy models are pictured in figure 6, along with their dependencies.

### The Tokenizer

The tokenizer is not present in the figure above, since it is considered different from the other components in that it receives a string as an input and returns a Doc object, while the others receive the Doc and return a processed version of it. Despite this, the tokenizer is an essential pipeline step, and it is the first to run, before the tok2vec component in the case of the en\_core\_web\_lg model.

The Doc returned by this pipeline element contains the tokens obtained by splitting the text into words, whitespaces and punctuation marks. The tokenizer treats special cases according to the language, for instance it splits “Let’s” or “don’t” into two tokens even though they are not delimited by a whitespace, while it considers acronyms like “U.K.” and “N.Y.” as single tokens.

O imagine care conține text, captură de ecran, Font, număr

Descriere generată automat

Figure 7: spaCy tokenizer

Figure 7, taken from spaCy’s official documentation (*SpaCy*, n.d.), shows how spaCy tokenizes a sentence: it splits it by whitespaces first, then it treats special cases such as exceptions (“Let’s” and “N.Y.”), prefixes (opening quotes) and suffixes (exclamation point, closing quotes) from left to right.

### The tok2vec

The token-to-vector component transforms the tokens into word vectors, numerical representations that encapsulate the semantics of words. Other components that rely on word embeddings can use the results made available by the tok2vec layer through listeners, as it can be seen in figure 6 for tagger, parser and lemmatizer. However, the named entity recognizer (ner) is intended to be independent of the other steps so that it can be used by itself, hence it contains its own tok2vec layer.

### The part of speech tagger

The POS tagger makes statistical predictions about the syntactic role of each token. The Token class contains two attributes that provide information about the part of speech: “pos\_” and “tag\_”, which goes into more detail. I will illustrate the tagger using the same sentence from figure 7.

Table 1: POS tags

|  |  |  |
| --- | --- | --- |
| TOKEN | POS | TAG |
| “ | PUNCT | punct |
| Let | VERB | VB (verb, base form) |
| ‘s | PRON | PRP (pronoun, personal) |
| go | VERB | VB |
| to | ADP (adposition) | IN (conjunction, subordinating or preposition) |
| N.Y. | PROPN (proper noun) | NNP (noun, proper singular) |
| ! | PUNCT | punct |
| “ | PUNCT | punct |

I obtained the results in table 1 by applying the spaCy pipeline on the chosen sentence and printing the pos\_ and tag\_ attributes of each token, as well as the explanation of what the abbreviations mean, using spacy.explain().

### The dependency parser

The dependency parser builds a dependency graph that models the syntactic structure of a phrase, where words are represented as nodes and the edges are labelled with the relationships between them.

O imagine care conține schiță, diagramă

Descriere generată automat

Figure 8: displaCy dependency visualizer

The following dependencies are pictured in figure 8, which I made using displaCy visualizer[[1]](#footnote-1):

* ccomp - clausal complement
* nsubj - nominal subject
* prep - prepositional modifier
* pobj - object of preposition

### The attribute ruler

The attribute ruler layer allows users to specify some rules for setting token attributes that may be incorrectly handled by other pipeline components such as POS tagger, ner or lemmatizer. This step is suitable in case of domain specific contexts or unusual uses of some words.

### The lemmatizer

The lemmatizer assigns base forms to tokens based on the POS tag, morphological rules and dictionary lookups. For instance, the lemma of “’s” token in the previously used sentence “Let’s go to N.Y.!” is “us”, while the other lemmas coincide with the original words.

### The named entity recognizer

The entity recognizer is a neural network model capable of annotating different types of proper nouns and numerical values with the following labels: CARDINAL, DATE, EVENT, FAC, GPE, LANGUAGE, LAW, LOC, MONEY, NORP, ORDINAL, ORG, PERCENT, PERSON, PRODUCT, QUANTITY, TIME, WORK\_OF\_ART (*SpaCy*, n.d.).

O imagine care conține text, captură de ecran, Font, număr

Descriere generată automat

Figure 9: displaCy entity visualizer [[2]](#footnote-2)

Figure 9 shows an example of a text with named entities labelled using spaCy.

According to the spaCy documentation, the part of speech tagger from en\_core\_web\_lg has an accuracy of 0.97 and its named entity recognizer has a precision of 0.85, a recall of 0.86 and an F-score equal to 0.85.

## fastText

fastText is a linear text classifier developed by Facebook AI Research (Joulin et al., 2016), that achieves an accuracy similar to that of deep neural networks classifiers, while being remarkably faster. The training of fastText lasts less than ten minutes for more than one billion words and the classification of 500,000 sentences takes less than one minute on an average multicore CPU.

fastText represents the text as a bag of n-grams in order to retain some information about the order of words, that the simple bag of words model does not consider. Words and n-grams are converted into word vectors.

During training, the probability distribution over classes is computed using hierarchical softmax. The classes are organized in a binary decision tree and the probability of a class is computed as the product of the probabilities of all the nodes visited during a depth first traversal from root to that leaf node which corresponds to a class. The purpose of training is to learn the configurations of matrixes A and B that minimize the loss between the predicted classes and the actual labels (Joulin et al., 2016) :

(5)

In formula (5), N is the number of documents, is a vector which indicates the correct label of document n ( if document n belongs in class k, otherwise ), is the normalized vector of n-gram features and f is the softmax function described above, thus is the probability distribution over classes for document n. The algorithm uses stochastic gradient descent to find the parameters A and B that minimize the value of the negative log-likelihood formula (5) (Joulin et al., 2016).

Joulin, Grave, Bojanowski, Douze, et al. (2016) described some model compression techniques that can drastically reduce the memory usage without greatly impacting the accuracy: product quantization (splitting the word vectors into smaller subvectors and then quantizing them using nearest neighbour), pruning the vocabulary (keeping only a fixed number of the most relevant words and n-grams), hashing trick and Bloom filter (hashing words and n-grams in order to reduce the size of the dictionary and using Bloom filters to check the presence of a term in the dictionary in constant time).

I used the Python fastText library for the language identification purpose[[3]](#footnote-3). It provides two models capable of recognizing 176 languages, one with a size of 126MB and the compressed version with a size of 917kB. I chose to use the larger model, because it offers slightly superior speed and accuracy.

## Gensim

## 

# Soluția propusă

Capitolul conține o privire de ansamblu a soluției ce rezolvă problema, prin prezentarea structurii / arhitecturii acesteia. În funcție de tipul lucrării acest capitol poate conține diagrame (clase, distribuție, workflow, entitate-relație), demonstrații de corectitudine pentru algoritmii propuși de autor, abordări teoretice (modelare matematică), structura hardware, arhitectura aplicației.

Criterii pentru calificativul *Nesatisfăcător*:

* Descriere în limbaj natural.

Criterii pentru calificativul *Satisfăcător*:

* Descriere + diagrame de baze de date, workflow, clase, algoritmi.

Criterii pentru calificativul *Bine*:

* Descriere + diagrame de baze de date, workflow, clase, algoritmi + descrierea unui proces prin care s-a realizat arhitectura/structura soluției.

## Indicații formatare formule

Formulele matematice utilizate în document vor fi centrate în pagină și numerotate. Se vor utiliza fontul Cambria Math, de dimensiune 11. Pentru a insera o nouă ecuație, utilizați Insert > Quick Parts > AutoText > Ecuație.

Toate formulele prezente în lucrare vor fi referite în text. Exemplu: *Utilizând sistemul de Insert > Bookmark*, respectiv *Reference > Cross-reference* putem cita ecuația (1) respectiv ecuația (2), citările fiind actualizate și în urma unor adăugări/ ștergeri de ecuații, cu *Select All – Update Field*. Pentru mai multe detalii despre utilizarea acestui sistem de referire și formatare puteți consulta:

<https://www.youtube.com/watch?v=9YGTH4WrY_8>.

|  |  |  |
| --- | --- | --- |
|  |  | () |

|  |  |  |
| --- | --- | --- |
|  |  | () |

# Detalii de implementare

În plus fata de capitolul precedent acesta conține elemente specifice ale rezolvării problemei care au presupus dificultăți deosebite din punct de vedere tehnic. Pot fi incluse configurații, secvențe de cod, pseudo-cod, implementări ale unor algoritmi, analize ale unor date, scripturi de testare. De asemenea, poate fi detaliat modul în care au fost utilizate tehnologiile introduse in capitolul 3.

Criterii pentru calificativul Nesatisfăcător:

* Sunt prezentate pe scurt scheme și pseudo-cod.

Criterii pentru calificativul Satisfăcător:

* Descriere sumara a implementării, prezentarea unor secvențe nerelevante de cod, scheme, etc.

Criterii pentru calificativul Bine:

* Descrierea detaliată a algoritmilor/structurilor utilizați; Prezentarea etapizată a dezvoltării, inclusiv cu dificultăți de implementare întâmpinate, soluții descoperite; (dacă este cazul) demonstrarea corectitudinii algoritmilor utilizați.

## Indicații formatare tabele

Se recomandă utilizarea tabelelor de forma celui de mai jos. Font: Calibri, 9.

Orice tabel prezent în teză va fi referit în text; exemplu: a se vedea Tabel 1.

Tabel 1 – Sumarizare criterii

|  |  |  |
| --- | --- | --- |
| Calificativ | Criteriu | Observații |
| Nesatisfăcător | Sunt prezentate pe scurt scheme și pseudo-cod |  |
| Satisfăcător | Descriere sumara a implementării, prezentarea unor secvențe nerelevante de cod, scheme, etc. |  |
| Bine | Descrierea detaliată a algoritmilor/structurilor utilizați; Prezentarea etapizată a dezvoltării, inclusiv cu dificultăți de implementare întâmpinate, soluții descoperite; (dacă este cazul) demonstrarea corectitudinii algoritmilor utilizați. | Pot fi incluse configurații, secvente de cod, pseudo-cod, implementări ale unor algoritmi, analize ale unor date, scripturi de testare. |

# Studiu de caz / Evaluarea rezultatelor

Acest capitol trebuie să răspundă, în principiu, la **2 întrebări** și să se încheie cu **o discuție** a rezultatelor obținute. Cele doua întrebări la care trebuie sa se răspundă sunt:

1) **Merge corect**? (Conform specificațiilor extrase în capitolul 2);

Evaluarea dacă merge corect se face pe baza cerințelor identificate în capitolele anterioare.

2) Cât de bine merge / cum se compară cu soluțiile existente? (pe baza unor metrici clare).

Evaluarea cât de bine merge trebuie să fie bazată pe procente, timpi, cantitate, numere, **comparativ cu soluțiile prezentate în capitolul 3**. Poate fi vorba de performanță, overhead, resurse consumate, scalabilitate etc.

În realizarea discuției, se vor utiliza tabele cu procente, rezultate numerice și grafice. În mod obișnuit, aici se fac comparații și teste comparative cu alte proiecte similare (dacă există) și se extrag puncte tari și puncte slabe. Se ține cont de avantajele menționate și se demonstrează viabilitatea abordării / aplicației, de dorit prin comparație cu alte abordări (dacă acest lucru este posibil). Cuvântul cheie la evaluare este „metrică“: trebuie să aveți noțiuni măsurabile și cuantificabile. În cadrul procesului de evaluare, explicați datele, tabelele și graficele pe care le prezentați și insistați pe relevanța lor, în următorul stil: „este de preferat ... deoarece …“; explicați cititorului nu doar datele ci și semnificația lor și cum sunt acestea interpretate. Din această interpretare trebuie să rezulte poziționarea proiectului vostru printre alternativele existente, precum și cum poate fi acesta îmbunătățit în continuare.

Criterii pentru calificativul *Nesatisfăcător*:

* Aplicația este testată dar rulează pe calculatorul studentului, nu există posibilități de testare, nu a fost validată cu clienți / utilizatori;
* Nu au fost realizate comparații cu alte sisteme similare.

Criterii pentru calificativul *Satisfăcător*:

* [Dezvoltare de produs] Există teste unitare și de integrare, există o strategie de punere în funcțiune (*deployment*), există validare minimală cu clienții / utilizatorii.
* [Cercetare] Principalele componente și soluția în ansamblu au fost evaluate din punct de vedere al performanței, însă nu sunt folosite seturi de date standard, există unele erori de interpretare a datelor.
* [Ambele] Discuție minimală asupra relevanței rezultatelor prezentate, comparație minimală cu alte sisteme similare.

Criterii pentru calificativul *Bine*:

* [Dezvoltare de produs] Teste unitare și de integrare, instrumente de punere in funcțiune (*deployment*) utilizate și care arată lucru constant de-a lungul semestrului, lucrare validată cu clienții / utilizatorii, produs în producție.
* [Cercetare] Componentele și soluția în ansamblu au fost evaluate din punct de vedere al performanței, folosind seturi de date standard și cu o interpretare corectă a rezultatelor.
* [Ambele] Discuție cu prezentarea calitativă și cantitativă a rezultatelor, precum și a relevanței acestor rezultate printr-o comparație complexă cu alte sisteme similare.

# Concluzii

În acest capitol este sumarizat întreg proiectul, de la obiective, la implementare, si la relevanta rezultatelor obținute. În finalul capitolului poate exista o subsecțiune de „Dezvoltări ulterioare“.  
Criterii pentru calificativul *Nesatisfăcător*:

* Concluziile nu sunt corelate cu conținutul lucrării;

Criterii pentru calificativul *Satisfăcător*:

* Concluziile sunt corelate cu conținutul lucrării, însă nu se oferă o imagine asupra calității și relevantei rezultatelor obținute;

Criterii pentru calificativul *Bine*:

* Concluziile sunt corelate cu conținutul lucrării, și se oferă o imagine precisa asupra relevantei și calității rezultatelor obținute în cadrul proiectului.

# REFERENCES

Abdelrazek, A., Eid, Y., Gawish, E., Medhat, W., & Hassan, A. (2023). Topic modeling algorithms and applications: A survey. *Information Systems*, *112*, 102131. https://doi.org/https://doi.org/10.1016/j.is.2022.102131

Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent dirichlet allocation. *Journal of Machine Learning Research*, *3*(Jan), 993–1022.

Campos, R., Mangaravite, V., Pasquali, A., Jorge, A. M., Nunes, C., & Jatowt, A. (2018). YAKE! Collection-Independent Automatic Keyword Extractor. *Information Retrieval Technology*, 806–810. https://doi.org/10.1007/978-3-319-76941-7\_80

Campos, R., Mangaravite, V., Pasquali, A., Jorge, A., Nunes, C., & Jatowt, A. (2020). YAKE! Keyword extraction from single documents using multiple local features. *Information Sciences*, *509*, 257–289. https://doi.org/10.1016/J.INS.2019.09.013

Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., & Harshman, R. (1990). Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, *41*(6), 391–407.

Griffiths, T. L., & Steyvers, M. (2004). Finding scientific topics. *Proceedings of the National Academy of Sciences*, *101*(suppl\_1), 5228–5235. https://doi.org/10.1073/pnas.0307752101

Jelodar, H., Wang, Y., Yuan, C., & Feng, X. (2017). *Latent Dirichlet Allocation (LDA) and Topic modeling: models, applications, a survey*.

Joulin, A., Grave, E., Bojanowski, P., Douze, M., Jégou, H., & Mikolov, T. (2016). FastText.zip: Compressing text classification models. *ArXiv Preprint ArXiv:1612.03651*.

Joulin, A., Grave, E., Bojanowski, P., & Mikolov, T. (2016). Bag of Tricks for Efficient Text Classification. *ArXiv Preprint ArXiv:1607.01759*.

Matthew Honnibal. (n.d.). *Dead Code Should Be Buried*. Retrieved June 9, 2024, from https://explosion.ai/blog/dead-code-should-be-buried

Moreno-Ortiz, A. (2024). *Keywords* (pp. 59–102). https://doi.org/10.1007/978-3-031-52719-7\_4

Nomoto, T. (2022). Keyword Extraction: A Modern Perspective. *SN Computer Science*, *4*. https://doi.org/10.1007/s42979-022-01481-7

Rose, S., Engel, D., Cramer, N., & Cowley, W. (2010). Automatic Keyword Extraction from Individual Documents. In *Text Mining: Applications and Theory* (pp. 1–20). https://doi.org/10.1002/9780470689646.ch1

*spaCy*. (n.d.). Retrieved June 9, 2024, from https://spacy.io/

Srinivasa-Desikan, B. (2018). *Natural Language Processing and Computational Linguistics: A practical guide to text analysis with Python, Gensim, spaCy, and Keras*. Packt Publishing Ltd.

Tijare, P., & Rani, P. (2020). Exploring popular topic models. *Journal of Physics: Conference Series*, *1706*, 12171. https://doi.org/10.1088/1742-6596/1706/1/012171

Zhang, C. (2008). Automatic keyword extraction from documents using conditional random fields. *Journal of Computational Information Systems*, *4*(3), 1169–1180.

# Anexe

1. https://demos.explosion.ai/displacy [↑](#footnote-ref-1)
2. https://demos.explosion.ai/displacy-ent [↑](#footnote-ref-2)
3. https://fasttext.cc/docs/en/language-identification.html [↑](#footnote-ref-3)