



**FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE**

ASSIGNMENT OF BACHELOR'S THESIS

Title: Error solution finder using external public services
Student: Jan Toušek
Supervisor: Ing. Jan Hnízdil
Study Programme: Informatics
Study Branch: Web and Software Engineering
Department: Department of Software Engineering
Validity: Until the end of summer semester 2019/20

Instructions

The aim of the bachelor thesis is to create a web API that returns a set of possible solutions from the available public external systems (Stack Overflow, MSDN, etc.) based on the inserted arbitrary error message.

1. Analyze current methods for key phrases extraction from text.
2. Research available external systems that provide information about solving errors.
3. Design and create a web API that combines the analyzed approaches from section 1 and returns possible solutions from chosen external systems from section 2.
4. Create a simple web application based on the API interface with following features:
 - 4.1 Enter a specified error message,
 - 4.2 Display the results clearly,
 - 4.3 Provide the API documentation.
5. Test implemented web application and API with test error datasets and evaluate used key phrases extraction methods.

References

Will be provided by the supervisor.

Ing. Michal Valenta, Ph.D.
Head of Department

doc. RNDr. Ing. Marcel Jiřina, Ph.D.
Dean

Prague October 22, 2018



**FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE**

Bachelor's thesis

Error solution finder using external public services

Jan Toušek

Department of Software Engineering
Supervisor: Ing. Jan Hnízdil

June 4, 2020

Acknowledgements

I would like to express my gratitude to my family and friends who supported me in my studies and in writing of this thesis. I would especially like to thank my supervisor Ing. Jan Hnizdil for his advice and guidance regarding this thesis.

Declaration

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

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

In Prague on June 4, 2020

.....

Czech Technical University in Prague

Faculty of Information Technology

© 2020 Jan Toušek. All rights reserved.

This thesis is school work as defined by Copyright Act of the Czech Republic. It has been submitted at Czech Technical University in Prague, Faculty of Information Technology. The thesis is protected by the Copyright Act and its usage without author's permission is prohibited (with exceptions defined by the Copyright Act).

Citation of this thesis

Toušek, Jan. *Error solution finder using external public services*. Bachelor's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2020.

Abstrakt

Řešení chyb, které vznikly v některé z aplikací používaných v dnešních společnostech, zabírá mnoho času. Díky různým internetovým zdrojům mohou vývojáři a správci těchto systémů jednoduše sdílet řešení k těmto chybám. Tato bakalářská práce se zaměřuje na vyhledání řešení takové chyby, pokud řešení existuje. Aby to bylo vůbec možné, jsou v této práci nejprve analyzovány systémy umožňující vyhledání informací a jejich vlastnosti. Poté jsou analyzovány existující metody pro extrakci klíčových částí textu, konkrétně z chybových hlášek. Nakonec jsou analyzovány systémy poskytující řešení různých chyb. V implementační části práce je poté popsána implementace systému, který je navržen na základě předchozí analýzy. Vytvořený systém by měl především šetřit čas strávený řešením chyby tím, že minimalizuje čas potřebný k vyhledání řešení na internetu.

Klíčová slova Extrakce klíčových frází, chyba, řešení, zpracování přirozeného jazyka, strojové učení, webová aplikace

Abstract

Solving errors that occurred in some system used in today's companies takes a lot of time. Thanks to different internet sources developers and system

administrators can easily share solutions to these errors. This bachelor thesis aims to find a solution to such error if the solution exists. To make this possible this bachelor thesis firstly analyses information retrieval systems and their properties. Secondly it analyses existing methods for key phrase extraction from text, specifically from error messages. Finally, it analyses systems that provide solutions to different errors. In the implementation part of the thesis the implementation of system which is designed based on previous analysis is described. Created system should primarily save time spent solving an error by minimizing the time needed to find a solution on the Internet.

Keywords Key phrases extraction, error, solution, natural language processing, machine learning, web application

Contents

Contents	ix
Introduction	1
Aims of the thesis	1
1 Analysis	3
1.1 Information retrieval systems	3
1.2 Natural language processing	8
1.3 Key phrases extraction	10
1.4 Evaluation of information retrieval system	12
1.5 Webs providing solutions	13
1.6 Error messages	14
1.7 Machine learning	16
1.8 Requirements	20
2 Design	21
2.1 Architecture	21
2.2 Used technologies	23
2.3 Machine learning model	24
2.4 Used key phrases extraction methods	28
3 Implementation	31
3.1 Development environment	31
3.2 SolutionFinderCore	32
3.3 SolutionFinderTagger	37
3.4 SolutionFinderAPI	45
3.5 SolutionFinderWeb	48
3.6 Reranking	49
4 Evaluation	51

4.1	Evaluation dataset	51
4.2	Edit distance from manually extracted key phrase	51
4.3	Solution relevance	53
4.4	Tagging effectiveness	54
4.5	Performance	55
4.6	User testing	55
	Conclusion	57
	Future improvements	57
	Bibliography	59
	A Used shortcuts	63
	B Contents of attached medium	65

List of Figures

1.1	Classification of retrieved documents	6
1.2	Precision vs. recall [2]	6
1.3	System function diagram [3]	7
1.4	Basic Web Search Engine Architecture and Process [4]	8
1.5	Tokenization of an example error message	9
1.6	Machine learning schema	17
1.7	Classification and regression problems [11]	18
2.1	Architecture of Solution Finder application	22
2.2	Machine learning model development process [14]	25
3.1	Directory tree of the application	32
3.2	Wrapper, Question and Tag objects from StackExchange API	34
3.3	Count of tags usage on StackOverflow site	38
3.4	Stackoverflow tags in question [21]	45
3.5	API documentation generated by Swagger	47
3.6	Example of finding a solution	48
3.7	Example output from solution finder	49
4.1	Edit distance between manually extracted key phrase, unmodified message and matched template	52
4.2	Average normalized edit distance from manually extracted key phrase for each method	53
4.3	Count of total and relevant solutions found for each method	54

List of Tables

3.1	tags.csv top 10 most used tags	37
3.2	AutoML all algorithms, reduced training dataset	43
3.3	AutoML selected algorithms, complete training dataset	43
3.4	Confusion table of the final trained model on test dataset	44

Introduction

Errors are an everyday part of processes running in many companies or institutions. An error can be caused by an error in data, error in the business logic of the process or by users mistake. When an error occurs, the respective error message is often automatically sent to an administrator of the information system or to the first line support team. The person responsible for the system should fix the error or delegate it to someone else. The person trying to solve an error either knows immediately where the problem is or tries to find the solution on the Internet.

The goal of this bachelor thesis is to create an application that extracts key phrase from the error message and tries to find the solution in online systems that provide solutions to errors. This applications task is to automate the process of searching for solution and enable the automat that sends notification about an error to include its solution. It should save time spent by searching for a solution and by analysis of the error message possibly better understand what the error is about and who should be chosen as the responsible person to solve it.

The result of this thesis is designed for companies that have more processes administrated by several different people. This application should simplify the work of IT employees in the first line, whose task it is to decide how severe the error is, which system it concerns and who should be assigned to solve it.

Aims of the thesis

The aim of the analysis part is to analyze current methods for key phrases extraction from text. One of the aims is to analyze information retrieval systems and the methods those systems use to search relevant documents. Another aim is to become acquainted with basic principles of using the key phrases extraction methods to extract important parts of text. Next aim is to analyze public external systems that provide information about solving errors

and the possibility of using them in proposed application. Last aim is to analyze user requirements for the resulting web application.

The aim of the implementation part is to design and implement web API, that will use the key phrases extraction methods analyzed in the research part. The API will use these methods to find a solution to given error using selected public external systems. Another aim is to create a web application that will use this API to provide user interface to search for solutions to error messages. Final aim is to test implemented API and web application and evaluate used key phrases extraction methods on test error dataset.

Analysis

The first chapter describes some of the basic concepts related to information retrieval systems. It also describes properties of these systems and methods these systems use for searching documents based on a user specified query. Furthermore, it presents methods used for natural language processing, key phrases extraction and selected information retrieval systems used by programmers when looking for a solution to an error they are trying to solve. Next section discusses some fundamental concepts of machine learning. In the end requirements for the proposed application are specified.

1.1 Information retrieval systems

One of the most basic information retrieval systems is a catalogue in a library, which helps people to find books by their author's name, ISBN etc. Today one of the most used information retrieval systems and systems that programmers and IT specialists usually use are web search engines. Web search engines are used worldwide to search information and almost everyone knows how to use them. In this section the way these engines work will be described in more detail.

1.1.1 Information retrieval

“Information retrieval (IR) is finding material (usually documents) of an unstructured nature (usually text) that satisfies an information need from within large collections (usually stored on computers).” [1]

The goal of this bachelor thesis is to create an application that finds solution to a given error on the internet. Therefore, it is an information retrieval system. However, unlike traditional information retrieval systems this application has no direct access to the collection of documents. Consequently, it must use a middleman in form of another information retrieval system - web

search engine. Some of the search engines used for this task are discussed in section 1.5.

1.1.2 Basic concepts in information retrieval

Document

Document is any form of data, upon which the information retrieval system is built. In the context of this bachelor thesis document will be a website.

Information need

The topic about which a person is searching for information. If someone needs to solve an error, their information need is the solution to this error.

Query

Query is the text that the user of an IR system uses to specify his information need. Specifying the correct query can sometimes be a difficult task even for humans. To get the best results possible in an automat, we need to extract the most important parts of an error message. This task is described in section 1.3.

Relevance

Document returned by an IR system is relevant, if it contains the information the user considers valuable according to his initial information need.

Precision

Precision is the number of retrieved relevant documents divided by the total number of retrieved documents.

$$Precision = \frac{\#ReturnedRelevantDocuments}{\#ReturnedDocuments}$$

Recall

Number of returned relevant documents divided by the number of all relevant documents.

$$Recall = \frac{\#ReturnedRelevantDocuments}{\#AllRelevantDocuments}$$

Effectiveness

The effectiveness of an IR system is connected to the ratio of precision and recall. The bigger the recall a system has the lower the precision and vice versa. [1]

As mentioned before, the proposed system will not have access to all the documents, so it would be difficult to measure **recall** in this specific case.

On the other hand, **precision** can be measured relatively easily. The user can always evaluate retrieved documents and specify which ones are relevant to him and satisfy his information need. This bachelor thesis aims to save time of the administrators responsible for correct functionality of a system. Therefore, precision is very important metric because users are only interested in relevant documents since inspecting irrelevant documents would only waste their time. For this reason, it is crucial to **maximize the precision**.

1.1.3 Errors in information retrieval

Precision and recall are linked to classification of errors in the collection of retrieved documents. A document retrieved by in IR system will be in one of four categories.

- **True positive** – document is relevant and was retrieved
- **True negative** – document is not relevant a was not retrieved
- **False positive (Type I error)** – document is not relevant and was retrieved
- **False negative (Type II error)** – document is relevant and was not retrieved

The goal of information retrieval systems is to maximize the ratio of true positives and true negatives and minimize the count of false positives and false negatives. This goal is characterized by the **precision/recall curve**, which shows the ratio between precision and recall for given IR system. An example of precision/recall curve is in Figure 1.2. Usually a compromise works for the precision and recall. For example, system which would return all documents for any query would have recall close to 1, however its precision would be close to 0 and vice versa.

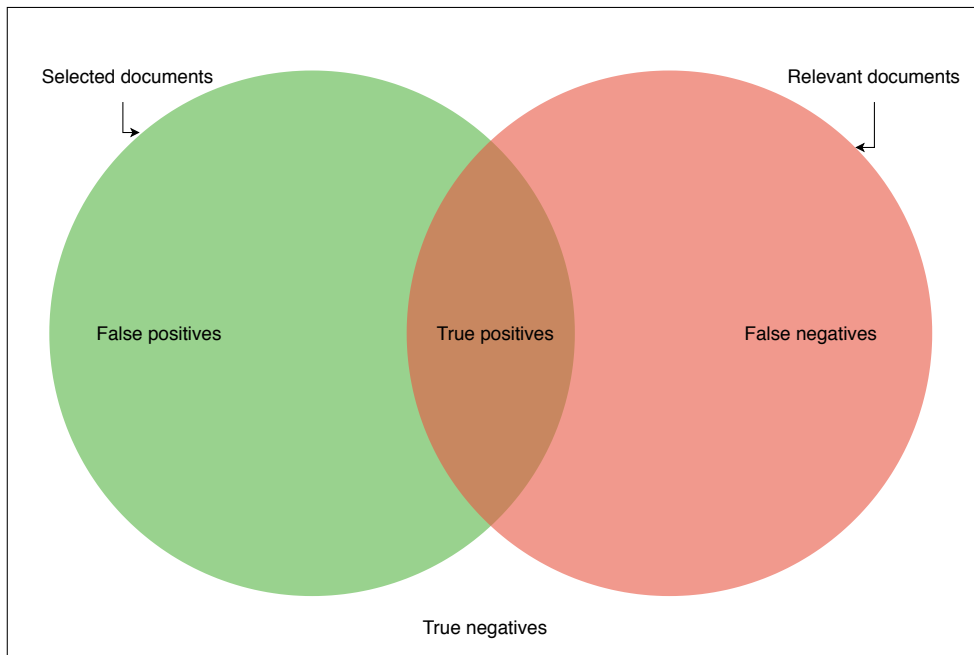


Figure 1.1: Classification of retrieved documents

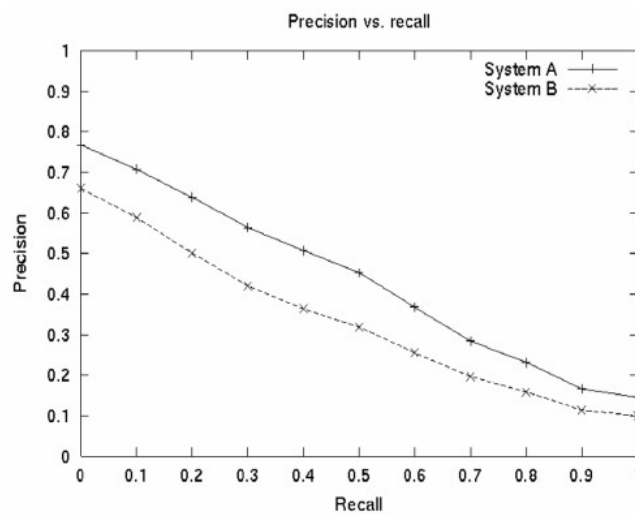


Figure 1.2: Precision vs. recall [2]

1.1.4 Existing solution

Chinese authors Fan Yang, Zhenghong Dong, Lihao Liu presented an interesting article named **Error Searching System with Keyword Extraction**

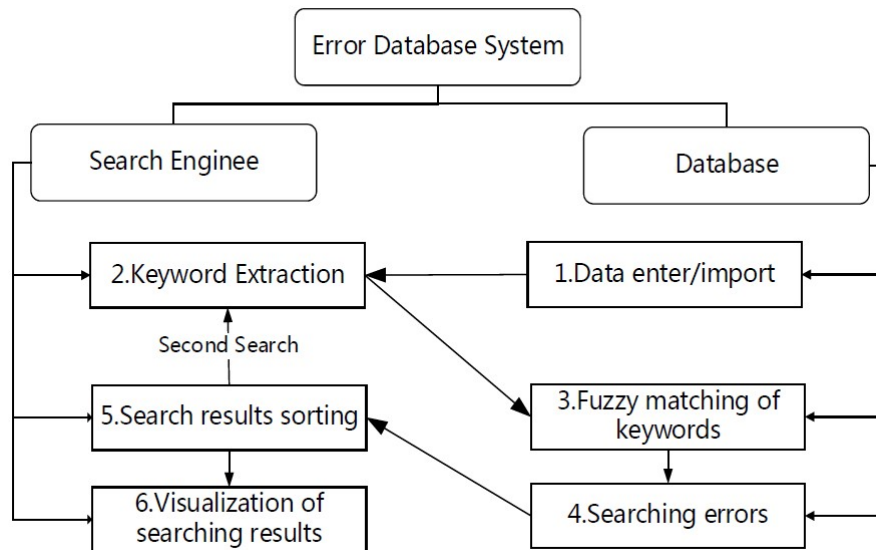


Figure 1.3: System function diagram [3]

and Keyword Fuzzy Matching [3]. They proposed and implemented a system focused on searching for solution to an error using key phrase extraction.

“Unified commanding platform mix-deployed software (UCPMD) integrates 22 sub-system from 4 different institutes, including 92 different software. Because of the differences of underlayer protocol and the differences of standard, there are many errors occurred during the stages of setup, configuration, and operation, which seriously affect the usage. Moreover, because those errors are various, which may be happened in different operation phases, stages, TCP/IP communication protocol layers, sub-system software, it is necessary to design a database system which can manage those errors.”[3]

One drawback of this system is the need to fill a solution to all errors, which had occurred, in one local database. If an error occurs for the first time it is necessary to create the solution manually. This bachelor thesis aims to remove this drawback by replacing one local database with module that is able to find the solution on the internet using a web search engine without the need to create the solution first.

1.1.5 Web search engines

”Web search engines are an important class of portals whose primary purpose is to support searches on a wide variety of topics across a comprehensive range of Web sites. Web search engines are a special form of information retrieval (IR) systems designed specifically for the hypermedia environment of the Web.” [4]

Web search engines with the leading market share as of January 2020 are Google (87.35%), Bing (5.53%) and Yahoo! (2.83%)[5]. These search engines usually use crawlers. Crawler is a program, that goes through the web using links on visited websites similarly to a user and returns the content and URL address of visited websites. Subsequent part of web search engine is indexer, which creates index from visited websites. Index is a database that contains URL address of a website and its content. This index is used later by the web search engine to search relevant websites according to users query.

Other search engines may only search websites inside one specific domain and their index contains only websites from that domain. When a user hits the search button, his query is at first split to terms and those terms are being searched in the index. Documents that contain specified terms are then returned to the user. For ranking the results search engines typically use similarity score, which based on various (usually proprietary) algorithms assigns score to each document. Documents are then ordered by this score.

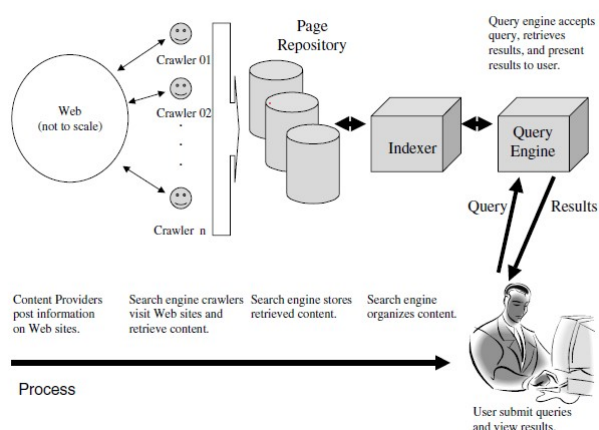


Figure 1.4: Basic Web Search Engine Architecture and Process [4]

1.2 Natural language processing

In this section some of the methods used for natural language processing and methods used by IR systems, for example web search engines, for processing user queries and searching for documents based on those queries are described.

- **Token** – sequence of characters in a document that carries some semantic information
- **Type** – class of all the tokens that consist of the same sequence of characters

- **Term** – token that is included in the vocabulary of IR system

1.2.1 Tokenization

Tokenization is a process, during which given document is split into individual tokens. Concurrently it is possible to remove some special characters, for example punctuation.

In ideal case the tokenization of the query and searched documents should be done by the same tokenizer in order to get the same tokens for same sequences of characters. In general tokenization is language specific. [1]

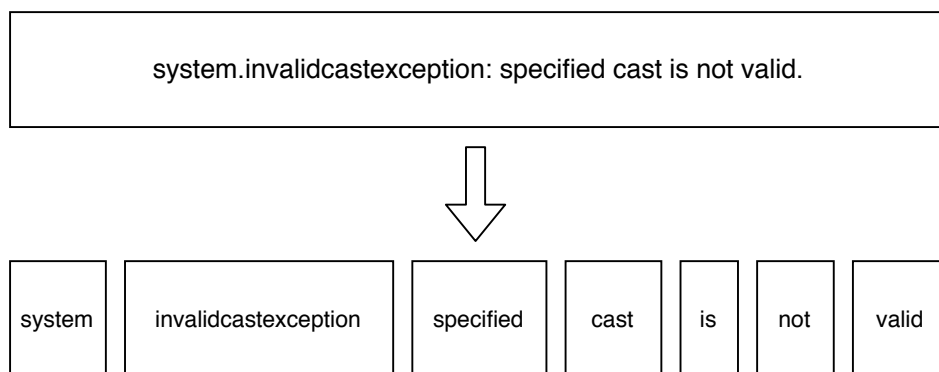


Figure 1.5: Tokenization of an example error message

1.2.2 Dropping common terms

Terms with little value for selecting matching documents, for instance because they are very common and appear in most of the documents, are usually removed from the dictionary of an IR system. These terms are called **stop words**. Stop words can be identified by comparing the frequency of occurrence of each term in the collection of documents.[1] Stop words are usually not indexed by the IR system and there is no reason in including them in the query.

1.2.3 Tagging

“Tags are descriptive terms users attach to online content, either their own or other user’s. Tagging is the practice of attaching tags. Tagging has been rapidly adopted on the Web, particularly by sites based on user-contributed content, such as blogs and photo sharing sites.” [6]

Tags can be used in many different ways but usually they are used to categorize, look up or filter content created by users on the Web. It is much easier for an IR system to work with tags compared to feature extraction from images, video or even text content.

1.2.4 Stemming and Lemmatization

Documents generally use different forms of the same word (cat, cats, cat's). It can be useful to search for all the words because they are very similar. "The goal of both stemming and lemmatization is to reduce inflectional forms and sometimes derivationally related forms of a word to a common base form."

Stemming is a heuristic process that removes prefixes or suffixes. Lemmatization on the other hand uses vocabulary and morphological analysis.

"The most common algorithm for stemming English, and one that has repeatedly been shown to be empirically very effective, is Porter's algorithm." [1]

1.2.5 Case-folding

Case-folding is often used when processing text. It means converting all characters to the same case, usually **lower case**. After the conversion it is possible to easily compare words, which only differed in case of one or more characters.

1.3 Key phrases extraction

In general error messages can be very long. There are two main problems with long queries. First problem is that some web search engines limit the maximum length of the query and therefore too long queries would have to be trimmed. Second problem is related to the way search engines work. They try to find a document in their index, that contains all the tokens in the query. If the query is too long such document in most cases does not exist.

Error messages often contain many different dynamic parts. Those dynamic parts are different according to the environment, time, user etc. For effective search of the solution it is needed to preprocess the error message before using it as a query. We need to select small number of tokens, which describe the content of the document. Those tokens are called key phrases. Key phrase is made of multiple key words.

1.3.1 Key phrase

Key phrase is a small set of tokens, that describes the content of the document well. Selection of the key phrase is usually done by a user based on his knowledge of the document. For automated search we need to automate the selection of key phrase as well.

1.3.2 Manual key phrase extraction

The task of extracting the key phrase from an error message is done daily by almost every programmer. It is done using his knowledge of the system and personal experience. However, there are also some patterns that are common

across a range of different systems. Defining and using these patterns to perform this task automatically will be useful.

1.3.3 Pattern detection

If it is possible to identify repeating patterns in the document collection, these patterns can be used to extract the key phrase.

1.3.4 Tf-idf

In vector model of information retrieval where the terms in a document are viewed as vectors the metric tf-idf was implemented for calculation of the weight of a term in a document. This metric is often used for evaluation if the term is a key word.

- **Term frequency**

tf_{ij} = count of occurrences of term t_j in document D_i

- **Inverse document frequency**

$$idf_j = \log\left(\frac{d}{df_j}\right)$$

where d is the number of documents

Calculation of the weight is as following:

$$d_{ij} = tf_{ij} * idf_j$$

Thus, the weight of the term increases if it occurs multiple times in a single document and decreases if it occurs in multiple documents.

1.3.5 Machine learning approaches

Given a large collection of documents that have assigned their key words it may be possible to train a machine learning model, which will predict models for new documents. However, machine learning based key phrases extraction methods are usually used on **different types of data**. Some of the most common usages include articles, abstracts or even books.

This approach was well described by Anette Hulth in her dissertation **Combining machine learning and natural language processing for automatic keyword extraction**.

Hulth pre-processes the data by extracting candidate terms. This is done by extracting uni-, bi and trigrams, excluding stop words like a, an and the and extracting empirically defined patterns. Thereafter, natural language processing methods like case-folding and stemming which are described in

section 1.2 are used. To create the model Hulth calculates features for each term. The features include term frequency, inverse document frequency and the position of first occurrence. These extracted features are then used to train a learning algorithm. This algorithm outputs a regression value. [12]

1.3.6 Removal of non-keywords

Another way to look at the problem of key phrase extraction is to remove the words that are not key. These words may include stop words (see subsection 1.2.2).

1.4 Evaluation of information retrieval system

One of the aims of this thesis is to evaluate used methods for key phrases extraction. This aim will be addressed in this section.

1.4.1 Quality of a retrieval system

Previously in this chapter various methods used by IR systems for searching relevant documents were described. We need to evaluate these methods and determine, whether it is possible to use them for the purposes of this bachelor thesis. In this section we will summarize ways of **evaluating effectiveness** of given method. The key is to satisfy users information need as closely as possible.

1.4.2 Test collection

To measure the effectiveness of an IR system we need test collection made of three objects:

- **collection of documents,**
- **test suite of information needs that can be translated to queries,**
- **set of relevance judgements.** [1]

Using these objects can be used to calculate the Precision and Recall.

1.4.3 Relevance feedback

Relevance feedback is a way of getting users feedback on the result of their search. After the results are retrieved, user has the option to evaluate results and tell if it was relevant for his information need. This feedback can be used later to improve the quality of IR system.

1.4.4 Extrapolation assumption

For well-chosen set of test data it is possible to assume that if a system works well for this small set of data it will also work well for an order of magnitude larger set of data. Extrapolation assumption is sometimes used when user interaction is needed or when training of machine learning models takes too long.

1.5 Webs providing solutions

One of the goals of the research part of this thesis is to analyze systems, that provide solutions for errors to programmers. These systems will be used as the source information retrieval system. When deciding which system to use the following factors should be considered.

- The quality of content,
- the existence and the quality of systems API and its documentation,
- costs - usage should be free.

1.5.1 StackExchange

StackExchange is a network of many websites, on which users can ask, answer and rate questions on various topics. Both questions and answers can be upvoted or downvoted by the community to rate their quality. The original poster can mark the best answer as accepted answer. These options work well for maintaining sustainable state of millions of questions and answers. The most popular site on StackExchange network is **StackOverflow**¹. It was created to help programmers solve their problems while developing and managing software. Large community is able to answer the vast majority of questions asked. One of the types of questions relate to runtime errors, which often contain an error message. Another useful site on this network may be **ServerFault**², which is focused on problems of server administrators.

StackExchange has the largest community of programmers and developers in the world.

- **3.5M** – Questions Asked,
- **4.7M** – pageviews,
- **12M** – comments. [7]

¹<https://stackoverflow.com/>

²<https://serverfault.com/>

StackExchange provides **well documented API**³, which allows an application to make all user actions. This API can be used as a source IR system in the context of this thesis. The API is free to use but limited to 10000 calls per day.

1.5.2 Microsoft developer network forums

Microsoft developer network forums MSDN⁴ is a forum focused on products from Microsoft. This forum works similarly to StackExchange, however it does not provide an API for convenient access to its resources. Its advantage is that it is supported by Microsoft. On the other hand the topics are limited to Microsoft products.

1.5.3 Google Custom Search JSON API

Google search is the most popular tool for searching information on the internet. Google provides Google Custom Search JSON API⁵ to support application access to the search engine. Since Google search indexes the whole web, this API could be used as a single interface for accessing data from various sources.

1.6 Error messages

To understand the error messages from the perspective of key phrase extraction, we need to describe how an error message usually looks like and how we can use that knowledge to get the key phrase.

In section 1.3 we analyzed the methods for key phrases extraction. Statistical methods based on Tf-idf are however used mostly for key phrases extraction from longer documents like articles or books as they work with the frequency of occurrence of terms in documents. Machine learning based methods using labeled training dataset require documents that have their key words assigned for them to be trained.

Error messages however are often short texts in which the key words are exactly once so the frequency of occurrence is not a good metric. Moreover, it is not possible to assign key words to sufficient number of documents for the purposes of machine learning. Methods mentioned above are therefore not well suited for key phrase extraction in this specific case.

On the other hand, error messages often contain similar information and have similar structure. This fact can be used to analyze them based on their structure. The structure depends on many variables and every programmer

³<https://api.stackexchange.com/docs>

⁴<https://social.msdn.microsoft.com/Forums/en-US/home>

⁵<https://developers.google.com/custom-search/v1/overview>

can create his own unique error message in different languages and with different key information.

For simplicity I will assume only error messages written in English language, that come from frameworks or applications, that are used by many users. I will assume, that only these messages have a potential solution on the internet.

Examples of error messages from different programming languages:

Java exception with stack trace

Exception in thread "main"

```
java.lang.NullPointerException
  at com.project.Book.getTitle(Book.java:16)
  at com.project.Author.getBookTitles(Author.java:25)
  at com.project.Bootstrap.main(Bootstrap.java:14)
```

SQL server job

```
Executed as user: NT SERVICE\SQLSERVERAGENT. Unclosed
quotation mark after the character string '
touseja4@fit.cvut.cz'. [SQLSTATE 42000] (Error 105)
Incorrect syntax near 'touseja4@fit.cvut.cz'. [
SQLSTATE 42000] (Error 102). The step failed.
```

MySql

```
ERROR 1045 (28000): Access denied for user
'bill'@'localhost' (using password: YES)
```

1.6.1 Dynamic parts of error message

As mentioned in section 1.3 error messages contain dynamic parts, that are specific for given run of the failed process. To get the key phrases of the error message we need to be able to identify and remove these dynamic parts. Some of the dynamic parts are:

- ID of the run, user, process
- date, eventually date and time of execution
- file path

- class path
- username, email
- IP address

1.7 Machine learning

Lately machine learning became very popular in various fields of computer science. In this section the term itself and some of the most important properties of systems that use machine learning will be described.

”Machine learning usually refers to the changes in systems that perform tasks associated with artificial intelligence (AI). Such tasks involve recognition, diagnosis, planning, robot control, prediction, etc.“ [8]

There are many scenarios and tasks for which machine learning can be useful. Some of the reasons to use a machine learning model introduced by Nilsson (1998) are listed below.

- Machine learning methods can extract some relationships between data, that are not obvious.
- It is difficult to define a task, but it is easy to provide examples with paired input and expected output.
- There is too many information for human to process it.
- New information is being added to the system and it would be necessary to constantly implement new rules. [8]

1.7.1 Training

The implementation of a machine learning model consists of combining training data with training algorithm and creating the model. The general schema of training and using a machine learning model is shown in Figure 1.6. There are two main types of the training process. These types can even be combined

Supervised learning - Each example in the training dataset is labeled.
Used to map input to output.

Unsupervised learning - The labels in training dataset are not known.
Used to identify structure in data.

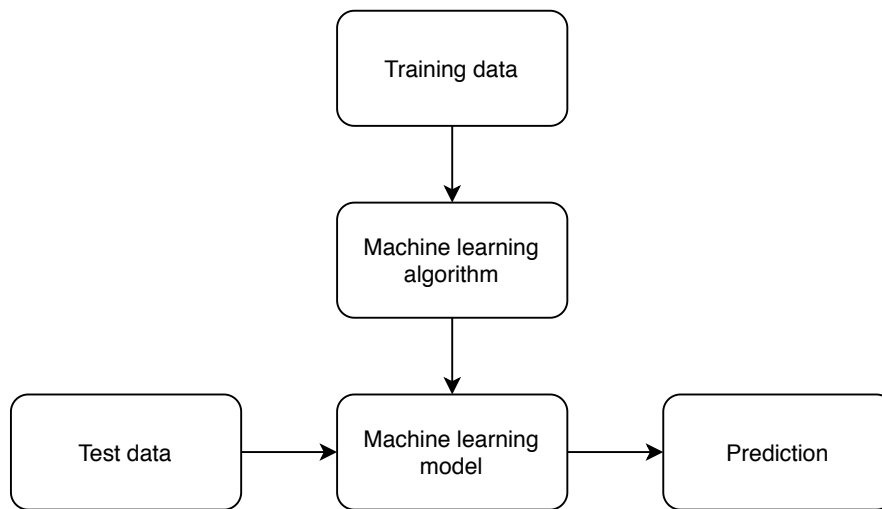


Figure 1.6: Machine learning schema

1.7.2 Data

The quality of data is very important for the overall quality of the machine learning model. In this bachelor thesis we will focus on the supervised learning training approach. Therefore a labeled dataset must be used to train the model. The datasets used for training must be big enough and the more complex the task, the bigger dataset is needed. The obtained data is usually divided into three datasets.

Training dataset - Used to train the model, should be the biggest.

Validation dataset - Used to tune the model.

Test dataset - Used to evaluate how well the model was trained. [9]

1.7.3 Machine learning problems

In this section some of the tasks that can be solved using machine learning are described.

Binary classification

Supervised learning method that predicts which of two classes given data belongs to. The input is labeled and each label has Boolean value 0 or 1. The output is also 0 or 1. [10]

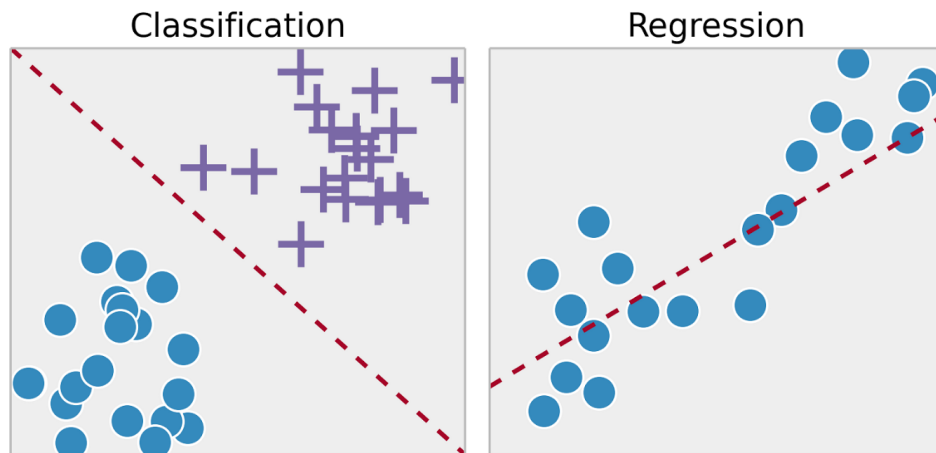


Figure 1.7: Classification and regression problems [11]

Multiclass classification

Supervised learning method that can be implemented using a decision tree. "A decision tree assigns a class number (or output) to an input pattern by filtering the pattern down through the tests in the tree. Each test has mutually exclusive and exhaustive outcomes." [8]

Regression

Supervised learning method that predicts the value of label from set of features. The output value is any real value and does not have to be from the set of values as in classification tasks. [10]

Clustering

Clustering is an unsupervised learning method, that uses some measure of similarity to group input data into clusters. The simplest approach would be to define a distance for example Euclidean distance between two points in n-dimensional space. [8]

1.7.4 Evaluation metrics of trained model

To evaluate the quality of a machine learning model several evaluation metrics are used. Some of them are described in this section. Metrics precision and recall were already mentioned in subsection 1.1.2.

F-measure

F-measure or F-score combines the precision and recall metrics. Closer to 1 means better predictions. The F-measure is calculated as following:

$$F_{\beta} = \frac{(1 + \beta^2) * Precision * Recall}{\beta^2 * Precision + Recall}$$

For $\beta > 1$ precision has more weight while for $\beta < 1$ recall has more weight in calculating the F-measure. When $\beta = 1$ the precision and recall have the same weight and the F1-measure is their harmonic mean. [1] The formula can be simplified to:

$$F_1 = \frac{2 * Precision * Recall}{Precision + Recall}$$

Accuracy

Accuracy is the fraction of correct predictions to the total number of predictions. [1] Closer to 1 means better predictions.

$$Accuracy = \frac{\#CorrectPredictions}{\#AllPredictions}$$

Log-loss

The log-loss uses the probabilities of predictions to evaluate the model. Closer to 0 means better predictions. Given p is the probability of prediction for the correct class i Log-loss is calculated as:

$$LogLoss(p_i) = -\log(p_i)$$

1.7.5 Use case for machine learning in this thesis

In this section machine learning was described in general. Further will be discussed two use cases for this specific topic.

Key phrase extraction

One possible usage of a trained machine learning model is the extraction of key words. This approach was described in subsection 1.3.5.

Tagging

Since the key phrase extraction approach presented by Hulth is quite complicated to implement, a simpler usage of a machine learning model will be implemented in this bachelor thesis. For a dataset having an error message and its tag it is possible to train a model, that can predict a tag for an unknown error message.

1.8 Requirements

Important part of system design is the definition of requirements. Requirements for the implemented system will be discussed in this section.

1.8.1 Functional requirements

- F1** - Find solution to an error - system must be able to find solution to an error when given an error message
- F2** - System must process given error message, assign correct tag and retrieve useful results
- F3** - For the purpose of tagging it is needed to train machine learning model, it must be possible to retrain this model any time automatically or manually

1.8.2 Non-Functional requirements

- N1** - Web application - system must be accessible as a web page, which can be viewed in modern web browsers
- N2** - Web API - system will provide documented REST application interface
- N3** - Optimization - web application should return results in a matter of seconds
- N4** - The result of the search will be clearly displayed on the web page

Design

2.1 Architecture

The application is made of four main parts. In this section the applications architecture will be described.

As shown in Figure 2.1 at first the user enters an error message using the website created in `SolutionFinderWeb`. After that the web application calls the `SolutionFinderAPI`. The API first uses the `SolutionFinderCore` library to get solutions and the correct tag for the specified error message. The `SolutionFinderCore` first uses the `SolutionFinderTagger` to get the tag and then uses an algorithm described in to get solutions for the error message. Both tag and found solutions are then returned to the API and then to web application where the user can view results.

Alternatively, the API can be called independently to provide another application the same results.

2.1.1 SolutionFinderCore

The main component of the application is the `SolutionFinder` module, which is built from the following parts.

- **StackOverflowClient** - communication with StackExchange API
- **Tagger** - uses trained machine learning model to assign tag to an error message
- **Key Phrases Extractor** - contains implemented methods for processing strings
- **SolutionFinderUtil** - the core of application, contains the algorithm for finding solution to an error

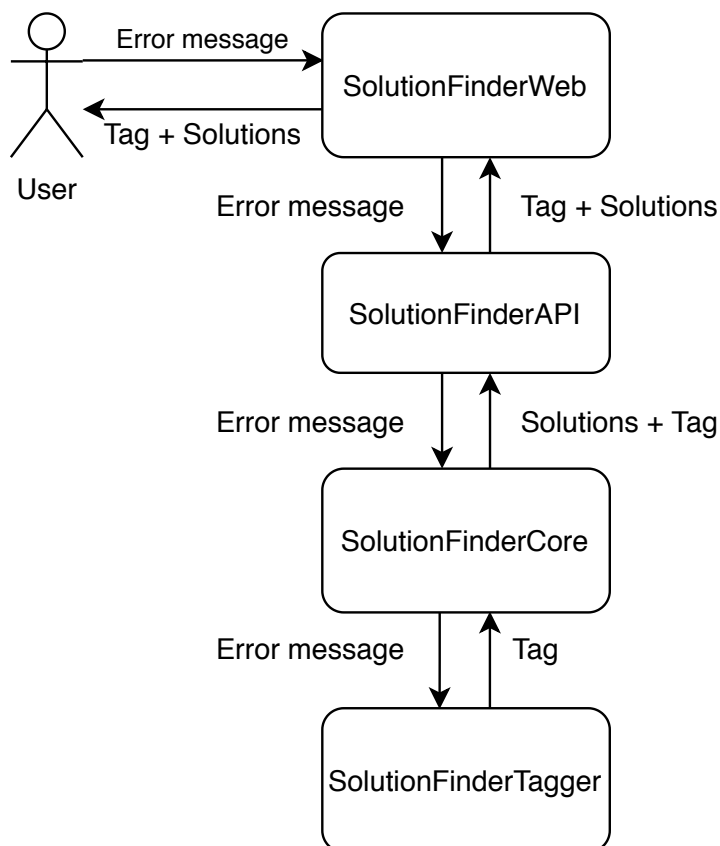


Figure 2.1: Architecture of Solution Finder application

- **Web Application** - enables the user to comfortably insert the error message and view retrieved results

2.1.2 SolutionFinderTagger

The SolutionFinderTagger module is responsible for assigning correct tag to an error message. To implement the tagging module two approaches were designed. Semantical and machine learning. However, soon it became obvious, that the machine learning approach gives much better results. Therefore, we will focus mainly on the machine learning approach. The design of used machine learning model is described in section 2.3.

Machine learning library

For the implementation of machine learning model, we have decided to use the ML.NET⁶ library written in .NET Core. "ML.NET is a free, cross-

⁶<https://docs.microsoft.com/cs-cz/dotnet/machine-learning/>

platform, open source machine learning framework made specifically for .NET developers.” [13]

2.1.3 SolutionFinderAPI

This module represents the server side of the application. It provides REST API which is then used by the client side application that is described in subsection 2.1.4. It provides an endpoint, that is used by the client to get solutions to given error message.

2.1.4 SolutionFinderWeb

The client side of the application. It should provide simple website, that enables users to enter an error message and view the search results clearly on the website. The client side needs to communicate with API described in subsection 2.1.3. The technology used to create the client must provide this functionality.

2.2 Used technologies

This section describes technologies used for implementation of the system.

2.2.1 Platform

Microsoft Azure⁷ is a cloud platform, that gives developers the option to deploy many different types of applications. It is natively compatible with .NET Core framework and it is very easy to run created applications. Among many other features it is possible to deploy web page, web API and container for application. [23] Microsoft Azure therefore provides the possibility to deploy all components implemented in this system.

2.2.2 Framework

For the implementation, the **.NET Core**⁸ framework was selected. Web sites and web APIs can be implemented using the **ASP.NET Core**⁹ framework which runs on .NET Core.

The framweork supports implementation of every part of the application presented in section 2.1:

- **library** – SolutionFinderCore described in subsection 2.1.1,

⁷<https://azure.microsoft.com/>

⁸<https://docs.microsoft.com/en-us/dotnet/core/>

⁹<https://docs.microsoft.com/en-us/aspnet/core>

- **machine learning model** – SolutionFinderTagger described in subsection 2.1.2,
- **web API** – SolutionFinderAPI described in subsection 2.1.3,
- **web site** – SolutionFinderWeb described in subsection 2.1.4.

The support of every module introduced in section 2.1 is one of the reasons to use this framework. .NET Core is free, cross-platform, open source and supported by Microsoft. [24] The framework is native for the Microsoft Azure platform. Project types of each of the components are:

- **SolutionFinderCore** – Class library (.NET Core)
- **SolutionFinderTagger** – Class Library (.NET Core)
- **SolutionFinderAPI** – ASP.NET Core Web Application
- **SolutionFinderWeb** – ASP.NET Core Web Application

2.2.3 Programming language

.NET Core supports C#, Visual Basic and F# programming languages. Based on personal preference and experience, C#¹⁰ was selected. C# is modern, object-oriented and type-safe programming language.

2.3 Machine learning model

The development of machine learning model is an iterative process. The process of training, evaluating and using the trained model to make predictions is visualized in the Figure 2.2.

2.3.1 Training of machine learning model

The ML.NET library supports various types of machine learning models. For example, Binary classification, Multiclass classification, Regression, Clustering etc. In this use case we need to assign the correct tag from predefined set of tags to given error message. It is therefore the **Multiclass classification** task.

The ML.NET documentation defines Multiclass classification task as following: “A supervised machine learning task that is used to predict the class (category) of an instance of data. The input of a classification algorithm is a set of labeled examples. Each label normally starts as text. It is then run through the TermTransform, which converts it to the Key (numeric) type. The output of a classification algorithm is a classifier, which you can use to predict the class of new unlabeled instances.”[13]

¹⁰<https://docs.microsoft.com/en-us/dotnet/csharp/>

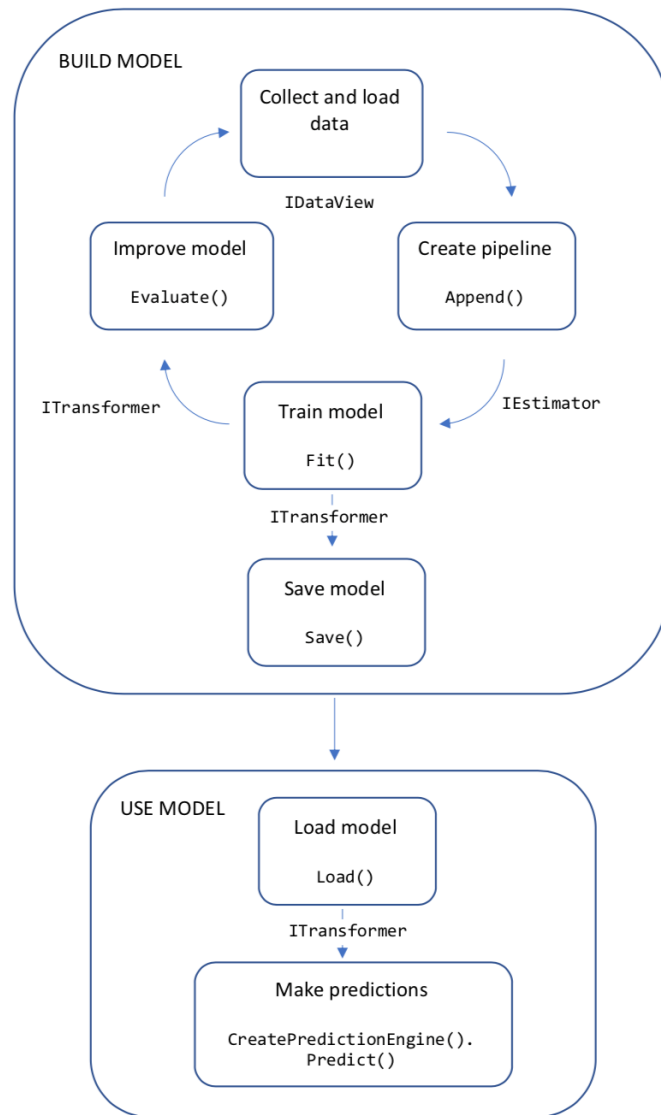


Figure 2.2: Machine learning model development process [14]

2.3.2 Multiclass classification trainers

The next step after selecting the type of task is the selection of training algorithm. For multiclass classification task the following algorithms are available:

Linear algorithms

The model produced by Linear algorithms is created by calculating scores from linear combination of the input data and a set of weights. The weights are estimated during training and passed as parameters to the model. Features should be normalized before training with a linear algorithm. Linear algorithms are usually fast and scalable.

- **Averaged perceptron** - best at text classification
- **Stochastic dual coordinated ascent** - good performance even without tuning
- **L-BFGS** - useful for large number of features, the output is logistic regression training statistic, does not scale well compared to Averaged perceptron
- **Symbolic stochastic gradient descent** - fastest, most accurate linear binary classification trainer, scales well [15]

Decision tree algorithms

The model created by a decision tree algorithm contains a series of decisions. Features do not need to be normalized. They do not scale as well as linear algorithms and use more resources.

- **Light gradient boosted machine** - fastest, most accurate (from binary classification tree trainers), highly tunable
- **Fast tree** - featurized image data, resilient to unbalanced data, highly tunable
- **Fast forest** - good for noisy data
- **Generalized additive model (GAM)** - for problems that are suited for tree algorithms but explainability is important [15]

Meta algorithms

These algorithms turn a binary trainer into a multiclass trainer.

- **One versus all** - one binary classification model for each class, this classifier differentiates that class from other classes,
- **Pairwise coupling** - trains a binary classification model for each combination of classes. [15]

2.3.3 Metrics

ML.NET uses the following metrics to evaluate model quality for multiclass classification tasks. Metrics are described in the documentation [16].

Micro-Accuracy

The fraction of correct predictions.

Macro-Accuracy

The average accuracy for all classes. classes of different sizes have the same weights.

Log-loss

The log-loss is bigger as the predicted probability diverges from the correct label.

Log-Loss Reduction

How much is the classifier better than a random prediction.

2.3.4 Cross-validation

Cross-validation is a technique that splits data into partitions for training and model evaluation. Afterwards it trains multiple algorithms on created partitions. By holding out data from the training process it improves the robustness of the model. It can also be effective for training models when the training dataset is small. [17]

2.3.5 AutoML

The ML.NET library contains **AutoML API**, which automates the process of training machine learning model using given dataset. Using the AutoML API it is possible to create an experiment, which will then use all available algorithms for training the model. It evaluates every used algorithm based on specified metric and selects the best one.

2.4 Used key phrases extraction methods

In this section selected methods for key phrases extraction will be described.

2.4.1 No modification

Sometimes the error message does not need to be processed. In that case the original string is used to find the solutions.

2.4.2 Error message templates

As mentioned in section 1.6, error messages often have similar structure. If the error message comes from the same system it is often possible to identify the key phrase based on its structure. Templates were implemented for some of the most used programming languages, like C#, Java or SQL. Below is an example of template implemented for SQL server errors and how it is used.

SQL server error

```
Executed as user: FITCVUT\jan.tousek.  
CREATE TABLE permission denied in database 'DWH'.  
[SQLSTATE 42000] (Error 262). The step failed.
```

Used regular expression

```
(?<=Executed as user:\s).*?\.\s(.*) (?=\s\[SQLSTATE[\s[A-Z\d]*\])
```

Output

```
CREATE TABLE permission denied in database 'DWH'.
```

2.4.3 Removal of dynamic parts

Dynamic parts were described in subsection 1.6.1. To remove these dynamic parts several methods were implemented. These methods use regular expression to remove the dynamic part from given error message. Regular expressions in .NET are a powerful tool to process text flexibly and efficiently. [18] The Listing 2.1 shows how numbers are removed from an error message using regular expression.

Implemented methods for removal of dynamic parts with examples

- **RemoveGuid** – 28d4d44f-8cda-420a-a1f2-575f5bd0a0c9
- **RemoveNumbers** – 42

```
public static string RemoveNumbers(string errorMessage)
{
    return Regex.Replace(errorMessage, @"[\d-]",
        string.Empty);
}
```

Listing 2.1: Remove numbers using regular expression

- **RemoveSpecialCharacters** – ?, !, %, :, ...
- **RemoveEmails** – touseja4@fit.cvut.cz
- **RemoveIP** – 147.32.232.212

2.4.4 Remove stop words

Removes all stop words using predefined list of English stop words.

2.4.5 Get first N tokens

For some error messages the most important information is located in the first few tokens. This approach returns first N tokens from the error message.

Implementation

3.1 Development environment

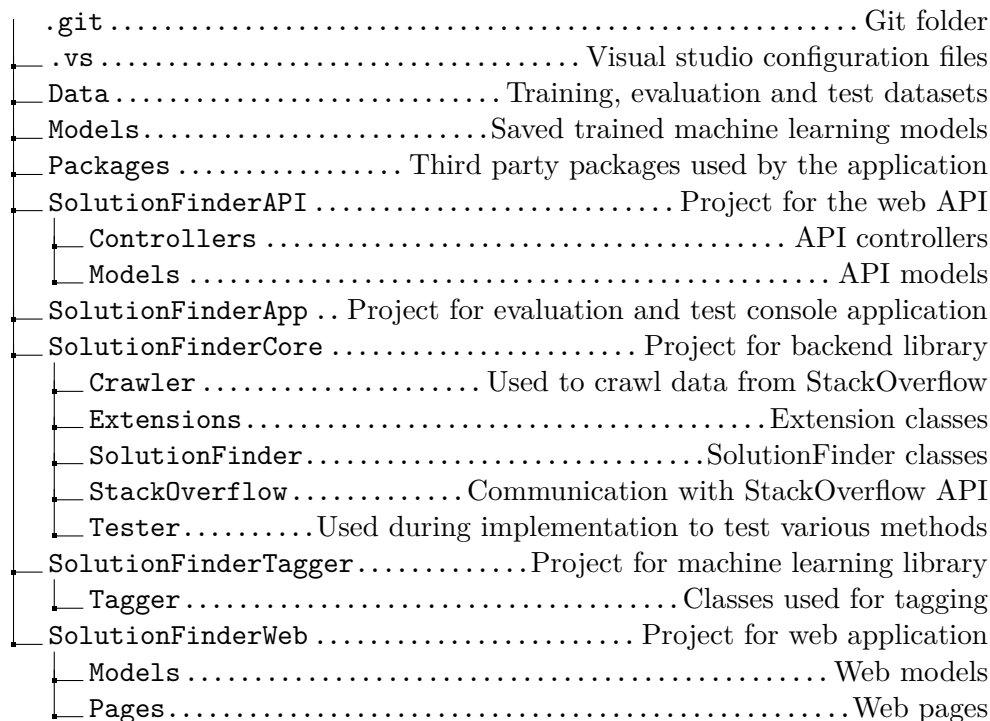
Visual Studio 2019 Community¹¹ was used as the development environment in this project. It is an integrated development environment (IDE) from Microsoft. Visual studio is used to edit, debug and build code. Compilers, code completion tools and other useful features are included. [19]

Directory tree of the solution

Directory tree of the solution is displayed in the Figure 3.1

¹¹<https://visualstudio.microsoft.com/cs/vs/>

Figure 3.1: Directory tree of the application



3.2 SolutionFinderCore

The SolutionFinderCore project represents the core of the application. It contains the client to communicate with StackExchange API and the methods used for the extraction of key phrases from error message.

3.2.1 StackOverflow client

The StackExchange API has some specific properties, which will be addressed in this section.

Response Wrapper

Responses from StackExchange API have the same format. Common wrapper object is returned. The most important fields in the wrapper are described below. Objects Wrapper, Question and Tag are displayed in the Figure 3.2.

- **backoff** - if set the application must wait the specified number of seconds
- **error_id** - only present when an error occurred during authentication or during processing the request

- **has_more** - used for paging
- **items** - array of objects returned in the request
- **quota_max** - maximum daily quota
- **quota_remaining** - actual daily quota remaining

Decompressing API Responses

All responses from the API are compressed with GZIP¹² or DEFLATE¹³ algorithms. According to the documentation: “The motivation for this is simple, serving uncompressed content is a loss for all parties. Bandwidth is, in comparison to CPU time, exceptionally expensive and severely limited on many devices. It’s really a no-brainer to require compression accordingly.” [20]

To properly consume the API response the **Accept-Encoding** header must be set. If not set, the GZIP will be used as default compression algorithm.

It is possible to configure the **HttpClient** to automatically decompress responses by providing **HttpClientHandler** with set **AutomaticDecompression** property as demonstrated in Listing 3.1.

```
var handler = new HttpClientHandler
{
    AutomaticDecompression = DecompressionMethods.GZip
};
var client = new HttpClient(handler);
```

Listing 3.1: Set AutomaticDecompression in HttpClient

Rate Limiting

There is a quota on the number of requests an application can send daily. By default, the quota is 10000 requests per day for all application on the same IP address. However, if **access_token** header is sent, the quota is distinct for user/application pair.

An **access_token** can be obtained by registering the application on Stack Apps¹⁴.

The API documentation also states that: “Every application is subject to an IP based concurrent request throttle. If a single IP is making more than 30 requests a second, new requests will be dropped.” [20] If the backoff field was set, the application must wait the specified number of seconds before making another request.

¹²<https://www.gzip.org/>

¹³<https://tools.ietf.org/html/rfc1951>

¹⁴<https://stackapps.com/apps/oauth/register>

3. IMPLEMENTATION

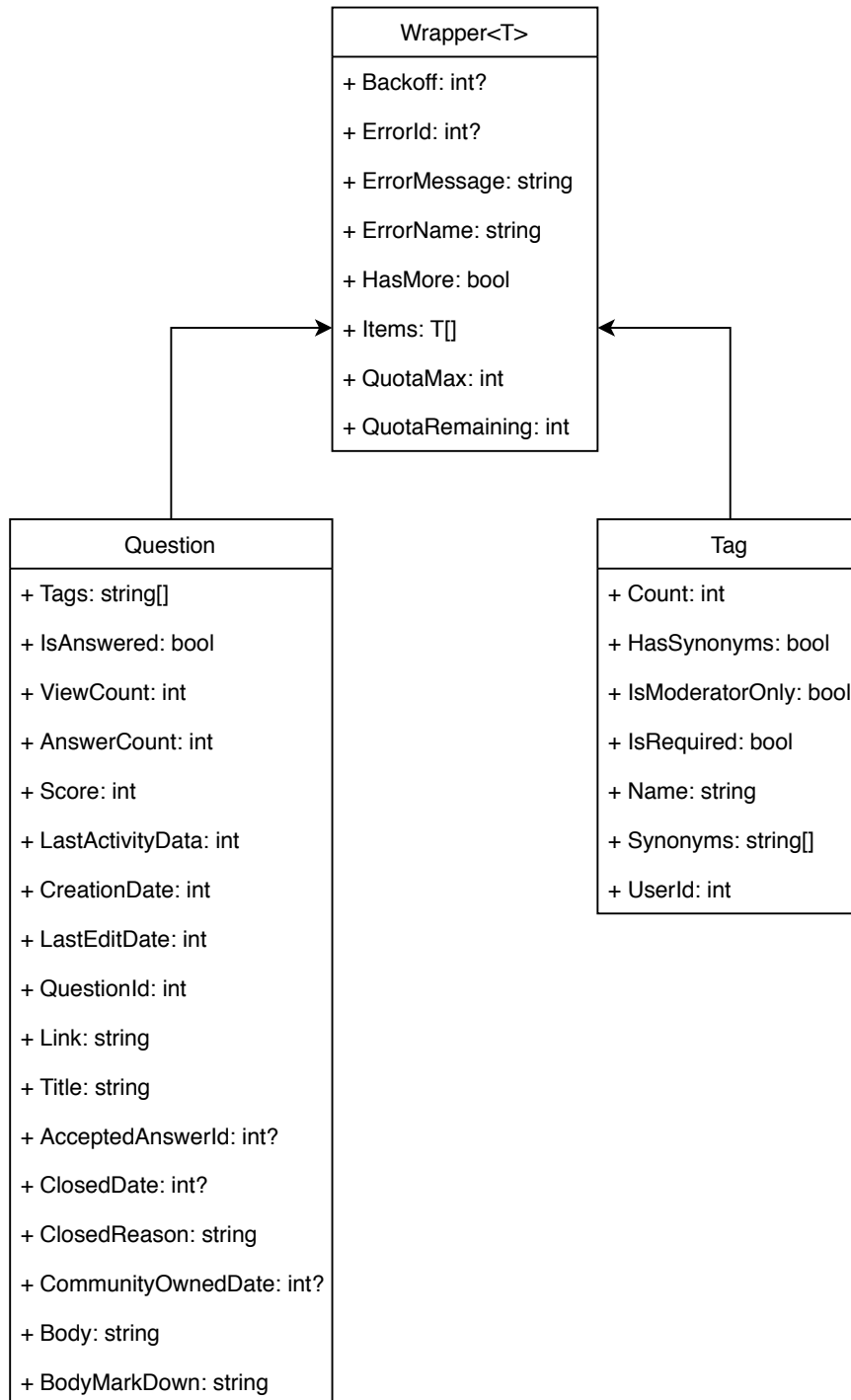


Figure 3.2: Wrapper, Question and Tag objects from StackExchange API

Filters

To allow the applications using the API to exclude returned fields filters are used. Filter can be created via the `/filter/create`¹⁵ method and then passed as a parameter in the request. Filters are used to exclude the fields that are not needed. This decreases the size of the response and therefore decreases the response time. Second use case is to include the fields, that are not present in the default filter. The use of a filter can be seen in Listing 3.2.

Paging

Most of the methods in the API accept `page` and `pagesize` parameters. If the application needs to fetch all items, the API returns the `has_more` field. The usage of this field is shown in Listing 3.2. The `GetTagsAsync` method is used to asynchronously return all tags on the StackOverflow site ordered by popularity and using a filter to only return necessary fields.

3.2.2 Crawler

This module is used to load data from StackExchange API and has two main purposes. Firstly it is used to get all used tags. Second purpose is to get as much questions, that contain an error message as possible.

GetTags method

This method uses the `GetTagsAsync` method shown in Listing 3.2 and saves tags on disc in CSV file `tags.csv`. The `/tags` method of StackExchange API returns the tags found on a site including their popularity - count of how many times the tag has been used on the site.¹⁶ Table 3.1 shows part of the generated file. The data in this file was used for the analysis of most used tags and the selection of tagging strategy.

GetQuestions method

For training the machine learning model it is necessary to get enough error messages for each of the selected tags listed in subsection 3.3.1. It is possible to get this dataset directly from the StackOverflow site because users who are looking for a solution to their error often include the whole error message in their question. Crawler implements method `GetQuestions` that uses the `/search` method in StackExchange API to get all questions that might

¹⁵<https://api.stackexchange.com/docs/create-filter>

¹⁶<https://api.stackexchange.com/docs/tags>

3. IMPLEMENTATION

```
public async Task<List<Tag>> GetTagsAsync()
{
    int page = 1;
    Wrapper<Tag> wrapper;
    var tags = new List<Tag>();

    do
    {
        var builder = new ApiUrlBuilder("tags");

        builder.AddParameter("order", "desc");
        builder.AddParameter("sort", "popular");
        builder.AddParameter("site", "stackoverflow");
        builder.AddParameter("filter", "!-*jbN*_Solyb");
        builder.AddParameter("page", page.ToString());
        builder.AddParameter("pagesize", "100");

        HttpResponseMessage response = await
            GetResponse(builder.ToString());

        wrapper = await GetWrapper<Tag>(response);
        tags.AddRange(wrapper.Items);

        page++;
    } while (wrapper.HasMore && page <= TAGS_PAGE_LIMIT);

    return tags;
}
```

Listing 3.2: Example usage of StackExchange API

contain an error message. The `GetQuestions` method saves received questions combined with their respective tags in CSV file `taggedQuestions.csv`.

3.2.3 SolutionFinder

The algorithm used to get solution for an error message is described in this section.

Processing error message

1. **Tagging** - trained machine learning model predicts tag for given error message
2. **Key phrase extraction** - various algorithms are used to extract key phrase from the error message

Table 3.1: tags.csv top 10 most used tags

Count	HasSynonyms	IsModeratorOnly	IsRequired	Name
1949497	True	False	False	javascript
1637413	True	False	False	java
1382438	True	False	False	c#
1350408	True	False	False	python
1333097	True	False	False	php
1252295	True	False	False	android
977458	True	False	False	jquery
968323	True	False	False	html
655303	True	False	False	c++
647952	True	False	False	css

3. **StackOverflow search** - extracted key phrases are used to find solution by querying StackExchange API
4. **Reranking** - found solutions are ordered

3.3 SolutionFinderTagger

StackExchange API provides endpoint to get tags for questions. At the same time it supports querying by tag. To get the best precision possible it is fitting to assign tag to each query. Tagging is implemented using the ML.NET library which was introduced in section 2.1.2.

3.3.1 Tag selection

There are thousands of different tags used on StackOverflow. Every user can create a new tag when asking a question. However, it does not make sense to train the machine learning model to recognize every tag. Training would take too much time and the model would not have sufficient training data for rarely used tags.

The count of uses of tags is displayed in the graph in Figure 3.3. As the graph shows, the distribution is very uneven. After deeper analysis some often used tags have been removed, since they are not directly connected to runtime errors e.g. HTML and CSS. Further some tags were merged into one, because they can be considered as synonyms. In the list below there are 10 selected tags, which will be used in training the machine learning model used for tagging error messages.

- **Javascript**
- **Java**

3. IMPLEMENTATION

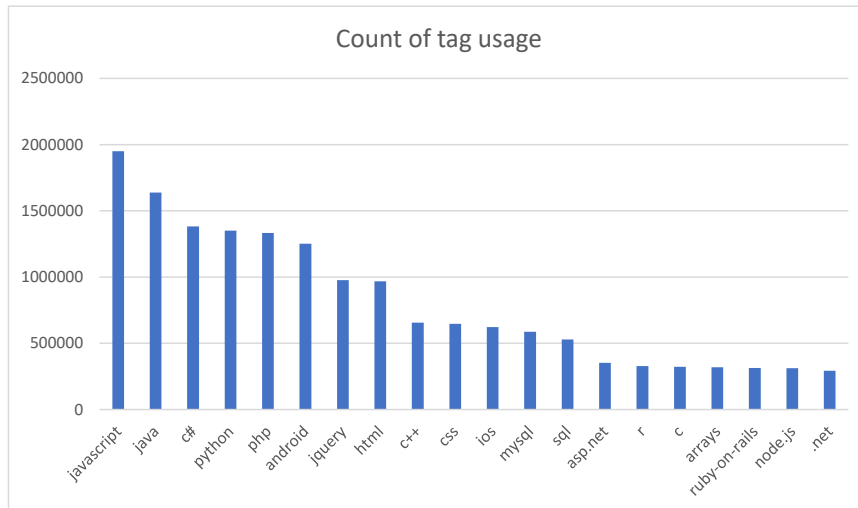


Figure 3.3: Count of tags usage on StackOverflow site

- **C#**
- **Python**
- **Php**
- **C++**
- **Sql**
- **C**
- **Ruby**
- **Objective-C**
- **Swift**
- **Linux**
- **Vba**

3.3.2 Training dataset

Training dataset for the machine learning model was created using the file `taggedQuestions.csv` described in subsection 3.2.2. However, there are many different types of asked questions and a large number of them does not contain the error message. Some questions are very long, they contain large amount of text written by the user who is asking the question, samples of code etc.

Nevertheless, neither the comments from user, nor code samples are relevant in the context of tagging an error message. For the aforementioned reasons questions need to be filtered only to those which contain an error message. The `body` property of `Question` object is in HTML. This became very useful for the extraction of error message from the question. The body for question in Figure 3.4 is displayed in Listing 3.3. [21]

```
<p>
  After installing <strong>Visual Studio 2012</strong>
  and opening my solution I get a series of errors in
  this form:
</p>
<blockquote>
  <p>
    The Web Application Project Foo is configured to use
    <strong>IIS</strong>.<br> Unable to access the
    <strong>IIS Metabase</strong>. You do not have
    sufficient privilege to access <strong>IIS</strong>
    web sites on your machine.
  </p>
</blockquote>
<p>
  I get this for each of our web applications.
</p><hr><p>Things I have tried:</p>
<ol>
  <li>Running Visual Studio as Administrator </li>
  <li>Running aspnet_regiis.exe -ga MyUserName </li>
  <li>Running aspnet_regiis.exe -i</li>
</ol>
<p>
  These seem to be common solutions for this problem
  but I have not had any success with them.
</p>
<blockquote>
  <p>
    Is there anything else I can try to do?
  </p>
</blockquote>
```

Listing 3.3: body of StackOverflow question in HTML

3. IMPLEMENTATION

After analysis of many questions related to an error, the following ways StackOverflow users use for highlighting an error message were discovered.

- **error: “error message”**

Text following “error:” or “exception:”

- **HTML tag blockquotes**

```
<blockquote>"the error message"</blockquote>
```

- **HTML tag code**

```
<code>"the error message"</code>
```

The error messages were processed using regular expressions and other conditions. The result dataset contains error messages and their respective tag. This dataset was used for training the machine learning model.

3.3.3 Preprocessing question for tagging

The dataset described in subsection 3.3.2 can be further processed to improve the performance and results of training. Below are the steps used for preprocessing the questions for tagging.

- **Remove HTML tags** - several HTML tags are used in StackOverflow questions, however only tags `blockquote` and `code` mentioned in subsection 3.3.2 have semantic importance. All other HTML tags can be removed. Removing unimportant HTML tags was done using library `HtmlAgilityPack`¹⁷
- **Remove whitespaces** - questions often contain large number of redundant whitespaces, which can be removed
- **CaseFolding** - we can convert all characters to lower case, more about the case folding can be found in subsection 1.2.5

3.3.4 Running AutoML experiment

The code sample Listing 3.4 shows how to create and run an AutoML experiment, how to set experiment settings, specify optimization metric and set the max time for experiment as `MaxExperimentTimeInSeconds`.

AutoML will train models as long as the training time is less than `MaxExperimentTimeInSeconds`. Further we can specify `CacheDirectory`, which is a directory that the library uses for caching trained models instead

¹⁷<https://html-agility-pack.net/>

of keeping them in memory. It is useful to specify `CacheDirectory` when training on large datasets because the models cached in memory can take a large amount of memory.

To run the experiment, we need to select the type of experiment in this case for multiclass classification problem we use `MultiClassClassificationTrainingExperiment`, training dataset and experiment settings. Optionally we can pass `ProgressHandler` function, which has a parameter of type `RunDetail<MulticlassClassificationMetrics>`, that contains the name of the current trainer, training time and optimization metric. `ProgressHandler` allows the observation of training in progress and display interim results.

“Running the experiment triggers data pre-processing, learning algorithm selection, and hyperparameter tuning. AutoML will continue to generate combinations of featurization, learning algorithms, and hyperparameters until the `MaxExperimentTimeInSeconds` is reached or the experiment is terminated.”[22]

Finally the best model stored in `experiment.BestRun.Model` is saved on disc in the `Environment.CurrentDirectory`, which is the current directory the application is running in.

3.3.5 Training using AutoML

The training time and `MicroAccuracy` metric for various training algorithms are significantly different. Training of the model is computationally intensive operation and for large training datasets it also requires large amount of memory. The selection of the training algorithm for the `SolutionFinderTagger` was done in two rounds.

First round

In the first round of training the experiment trains models with all available multiclass classification trainers. A reduced dataset was used to speed up the process of training. Purpose of this step is to select training algorithms that perform best for this task. The micro-precision measure was used to evaluate models in this step. Second important measure is the training time in seconds. Results of first round training are displayed in Table 3.2

Second round

Only the algorithms that had good `MicroAccuracy` and their training time was not too long were used in the second round with the complete training dataset. Results of second round training are displayed in Table 3.2

3. IMPLEMENTATION

```
public void RunAutoML()
{
    IDataView data = _mlContext.Data
        .LoadFromTextFile<Test>(
            _trainDataPath,
            separatorChar: ';',
            hasHeader: true);

    var settings = new MulticlassExperimentSettings()
    {
        OptimizingMetric =
            MulticlassClassificationMetric.MicroAccuracy
        MaxExperimentTimeInSeconds = 7200
    };

    var experiment = _mlContext
        .Auto()
        .CreateMulticlassClassificationExperiment(settings)
        .Execute(data, progressHandler: Progress);

    _mlContext.Model.Save(
        experiment.BestRun.Model,
        data.Schema,
        Path.Combine($"{Environment.CurrentDirectory}",
            "AutoModel.zip"));
}
```

Listing 3.4: Create and run AutoML experiment

3.3.6 Final trained model

As the final training algorithm, the `AveragedPerceptronOva` trainer mentioned in subsection 2.3.2 was selected. It has good `MicroAccuracy` and its training time is relatively short.

Cross validating the model

The model was validated using cross-validation (see subsection 2.3.4) using 5 folds, which splits data into 5 partitions. Average metrics of the evaluation are listed below. These metrics were described in subsection 2.3.3.

- **Micro accuracy** = 0.808
- **Macro accuracy** = 0.798
- **Log loss** = 0.634
- **Log loss reduction** = 0.718

Table 3.2: AutoML all algorithms, reduced training dataset

Algorithm	Duration	Micro accuracy
SgdCalibratedOva	47,421s	0,758
LightGbmMulti	776,141s	0,749
LbfgsLogisticRegressionOva	59,361s	0,749
SdcaMaximumEntropyMulti	101,863s	0,738
FastTreeOva	1924,836s	0,736
AveragedPerceptronOva	26,090s	0,732
LinearSvmOva	19,603s	0,714
SymbolicSgdLogisticRegressionOva	29,685s	0,677
FastForestOva	1394,3s	0,686
LbfgsMaximumEntropy	16.662s	0.108

Table 3.3: AutoML selected algorithms, complete training dataset

Algorithm	Duration	Micro accuracy
AveragedPerceptronOva	69.470s	0,811
SgdCalibratedOva	92,388s	0,802
LbfgsLogisticRegressionOva	177,251s	0,794
LightGbmMulti	1766,852s	0,790
SdcaMaximumEntropyMulti	49,216s	0,769

Further the model was evaluated by splitting the data into training and test datasets. Test dataset contains 20 % of all data. The confusion table of the final model is displayed in Table 3.4.

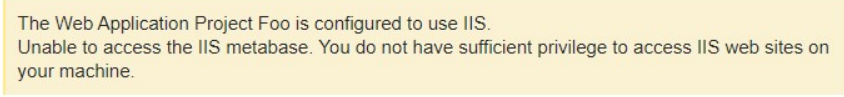
Table 3.4: Confusion table of the final trained model on test dataset

PREDICTED TRUTH	c#	c++	c	javascript	java	mysql	php	python	ruby	sql	Recall
c#	4561	47	13	81	74	61	53	38	14	249	0.8786
c++	74	2353	212	18	39	18	17	30	11	24	0.8416
c	30	252	957	23	24	11	13	47	20	15	0.6875
javascript	76	11	7	2136	55	14	74	42	20	18	0.8708
java	92	25	18	52	4032	129	26	31	7	108	0.8920
mysql	129	14	7	24	182	2275	473	85	38	257	0.6530
php	60	9	8	75	29	542	2758	29	25	57	0.7678
python	48	35	49	51	37	44	38	3448	20	44	0.9040
ruby	24	8	19	37	18	29	21	32	2091	25	0.9076
sql	293	7	8	17	133	706	122	42	14	1909	0.5872
Precision	0.8467	0.8522	0.7373	0.8496	0.8722	0.5941	0.7672	0.9017	0.9252	0.7055	

Error - Unable to access the IIS metabase

Asked 7 years, 4 months ago Active 5 months ago Viewed 355k times

▲ After installing Visual Studio 2012 and opening my solution I get a series of errors in this form:

838  The Web Application Project Foo is configured to use IIS. Unable to access the IIS metabase. You do not have sufficient privilege to access IIS web sites on your machine.


★ 160 I get this for each of our web applications. Things I have tried:

1. Running Visual Studio as Administrator
2. Running `aspnet_regiis.exe -ga MyUserName`
3. Running `aspnet_regiis.exe -i`

These seem to be common solutions for this problem but I have not had any success with them. Is there anything else I can try to do?

`c#` `.net` `iis`

share edit flag

edited May 9 '14 at 18:45  leppie 104k ● 16 ● 179 ● 282


asked Oct 12 '12 at 13:15  jjathman 11.9k ● 7 ● 26 ● 33

Figure 3.4: Stackoverflow tags in question [21]

3.3.7 Using trained model

After the model was trained it is possible to use it to make predictions. To make predictions with saved model we need to load it into memory, create an instance of `PredictionEngine` and call its `Predict` method. The method used for predicting a tag for an error message is displayed in Listing 3.5.

3.4 SolutionFinderAPI

This project implements REST application interface which is used by the web application to display retrieved data. Currently two endpoints are implemented. Documentation for this API was created using the Swashbuckle library.

GET /api/solution?errorMessage=

This endpoint is used to retrieve the solutions to a given `errorMessage` which is passed as parameter. The returned object `SolutionFinderOutput` contains the following fields:

- `TagMachineLearning`

3. IMPLEMENTATION

```
public static string PredictTag(string errorMessage)
{
    var trainedModel = LoadModel(_modelPath, _mlContext);

    var predictionEngine = _mlContext
        .Model
        .CreatePredictionEngine<Question,
            QuestionPrediction>
            (trainedModel);

    var prediction = predictionEngine.Predict(
        new Question
        {
            Message = errorMessage
        });
    return prediction.Label;
}
```

Listing 3.5: Create and run AutoML experiment

- TagSemantical
- Solutions
- SolutionFinderSteps
- TimeElapsed

GET /api/tag?errorMessage=

Returns TaggedError object for given errorMessage. This object contains:

- ErrorMessage – original error message
- TagMachineLearning
- TagSemantical

API Documentation

The Swagger documentation is available on the /swagger endpoint. Example of the generated documentation is displayed in Figure 3.5.

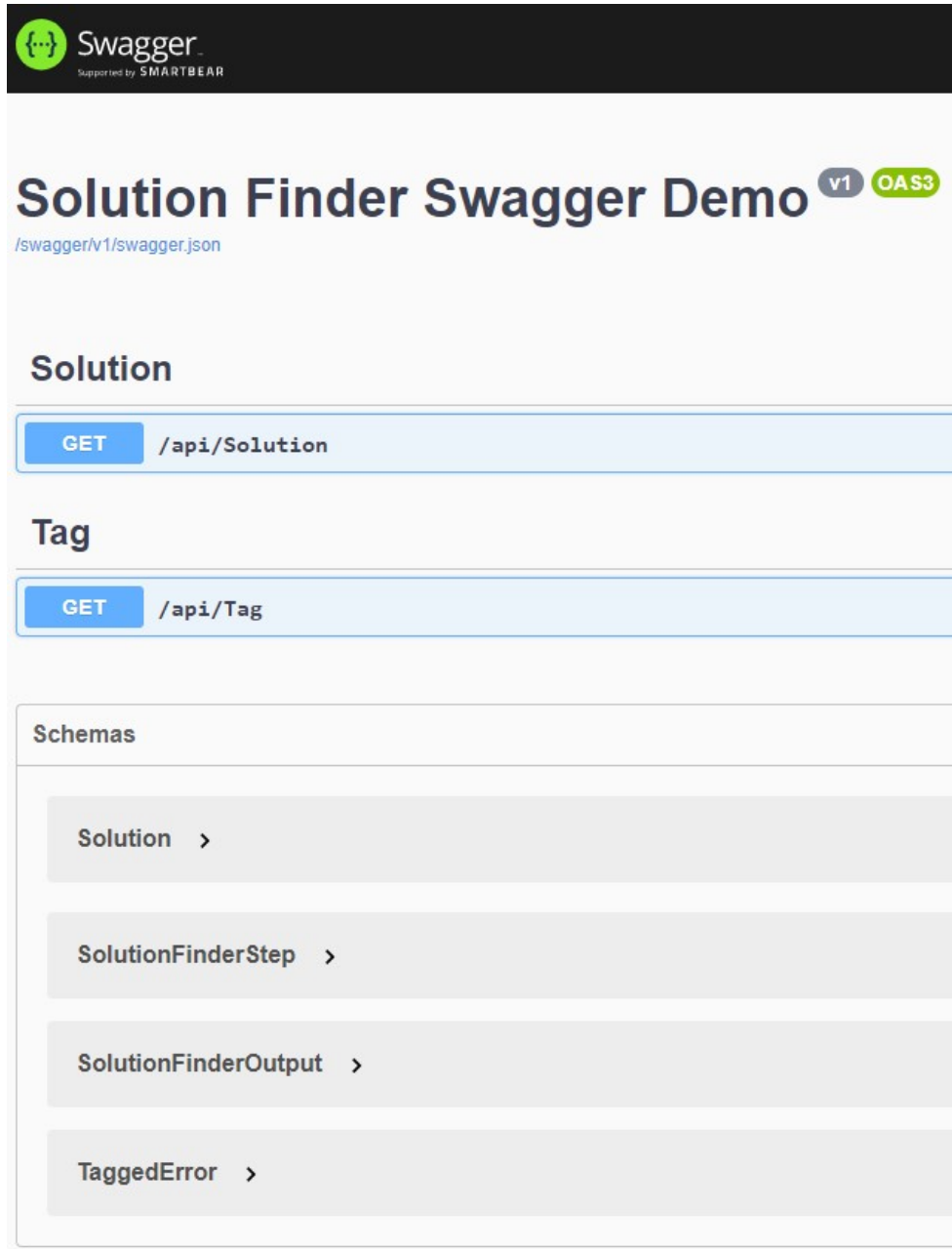


Figure 3.5: API documentation generated by Swagger

Solution Finder

Error Message:

```
Executed as user: FITCVUT\jan.tousek. CREATE TABLE  
permission denied in database 'master'. [SQLSTATE 42000]  
(Error 262). The step failed.
```

Find solution

Figure 3.6: Example of finding a solution

3.4.1 Deployment

To deploy the API the .NET Core 3.1 SDK is needed. The SDK can be downloaded on this website <https://dotnet.microsoft.com/download/dotnet-core/3.1>

From the application root directory go to the SolutionFinderAPI directory

```
cd .\SolutionFinderAPI
```

To build and run the API in Release configuration use

```
dotnet run --configuration Release
```

The API is running on port 6060 (default <http://localhost:6060>)

3.5 SolutionFinderWeb

This section concerns the development of the client side of the application.

3.5.1 Website

The website has one form that enables users to insert an error message and get solutions for it by clicking the Find solutions button. Example usage of this form is displayed in Figure 3.6.

After the request is processed by SolutionFinderAPI the output is displayed on the same page. Example output is displayed in Figure 3.7.


```

TagML: sql
TagSemantical: sql
TimeElapsed: 5041
Solutions:
  create table permission denied in database &#39;master&#39; Score: 12
  SQL Server 2008 - permission denied in database &#39;master&#39; Score: 4
  AWS MS SQL create table Score: 0
  Azure SQL Server Can&#39;t create table Score: 0
  How to import existing mdf file into a lightswitch project? Score: 0

Methods:
Method name: Unmodified
Message before: Executed as user: FITCVUT\jan.tousek. CREATE TABLE permission denied in database 'master'. [SQLSTATE 42000] (Error 262). The step failed.
Message after: Executed as user: FITCVUT\jan.tousek. CREATE TABLE permission denied in database 'master'. [SQLSTATE 42000] (Error 262). The step failed.

Method name: Template SQL
Message before: Executed as user: FITCVUT\jan.tousek. CREATE TABLE permission denied in database 'master'. [SQLSTATE 42000] (Error 262). The step failed.
Message after: CREATE TABLE permission denied in database 'master'.

Solutions from method:
  create table permission denied in database &#39;master&#39; Score: 12
  AWS MS SQL create table Score: 0
  SQL Server 2008 - permission denied in database &#39;master&#39; Score: 4
  Azure SQL Server Can&#39;t create table Score: 0
  How to import existing mdf file into a lightswitch project? Score: 0

Method name: RemoveAllVariables
Message before: Executed as user: FITCVUT\jan.tousek. CREATE TABLE permission denied in database 'master'. [SQLSTATE 42000] (Error 262). The step failed.
Message after: Executed as user: FITCVUT\jan.tousek. CREATE TABLE permission denied in database master. SQLSTATE Error. The step failed.

Method name: RemoveGuidNumbers
Message before: Executed as user: FITCVUT\jan.tousek. CREATE TABLE permission denied in database 'master'. [SQLSTATE 42000] (Error 262). The step failed.
Message after: Executed as user: FITCVUT\jan.tousek. CREATE TABLE permission denied in database 'master'. [SQLSTATE ] (Error ). The step failed.

```

Figure 3.7: Example output from solution finder

3.5.2 Deployment

To deploy the web application the .NET Core 3.1 SDK is needed. The SDK can be downloaded on this website <https://dotnet.microsoft.com/download/dotnet-core/3.1>

From the application root directory go to the SolutionFinderWeb directory

```
cd .\SolutionFinderWeb
```

To build and run the web application in Release configuration use

```
dotnet run --configuration Release
```

The web application is running on port 7070 (default <http://localhost:7070>). API url address must be specified in the appSettings.json file, property `BaseUrl`. Structure of the appSettings.json file is shown in Listing 3.6. The value is then injected into the `PageModel`.

3.6 Reranking

In subsection 1.1.5 the ranking of retrieved results was mentioned. Since reranking can be very useful to show the best results on top, simple reranking

3. IMPLEMENTATION

```
{
  "AppSettings": {
    "BaseUrl": "http://localhost:6060/api/"
  },
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft": "Warning",
      "Microsoft.Hosting.Lifetime": "Information"
    }
  },
  "AllowedHosts": "*"
}
```

Listing 3.6: SolutionFinderWeb appsettings.json

was implemented. Two factors were considered when reranking the results. A combination of these two factors gives us the measure which we can use to rerank the results.

How much was the error changed before querying

The first score is based on how many information we needed to cut off from the error message to get a result. The less changes, the higher score. Assigned based on used key phrases extraction method.

The popularity score on StackOverflow site

Thanks to the possibility to upvote or downvote questions we can use the score of the question to determine its quality based on the feedback by StackOverflow community.

Evaluation

This chapter describes the evaluation of used methods for key phrases extraction and evaluation of user testing.

4.1 Evaluation dataset

To make the evaluation possible, 50 error messages distributed equally in all tags were selected. For these error messages key phrase was extracted manually and compared to implemented key phrases extraction methods.

4.2 Edit distance from manually extracted key phrase

Edit distance or Levenshtein distance between two strings s_1 and s_2 the minimum number of edit operations needed to transform s_1 into s_2 . Allowed edit operations are:

- insert character,
- delete character,
- replace character. [1]

4.2.1 MatchTemplates

In the Figure 4.1 the MatchTemplate method is compared to Unmodified. If the template did not match the unmodified error message was used. The graph shows significant improvement in context of edit distance to manually extracted key phrase for error messages that fall into one of templated categories.

4. EVALUATION

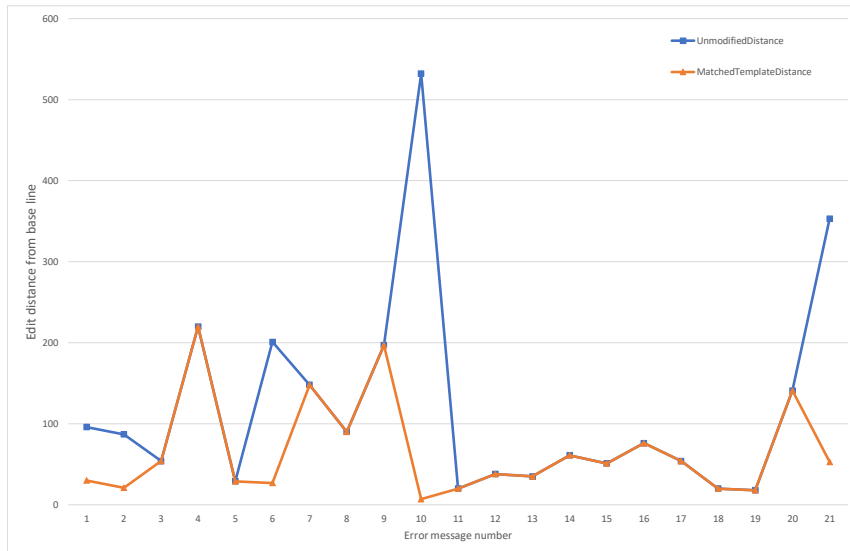


Figure 4.1: Edit distance between manually extracted key phrase, unmodified message and matched template

4.2.2 Normalized edit distance

Edit distance is an absolute value and therefore to compare the effectivity of methods used on error messages of different lengths it needs to be normalized. The edit distance in Figure 4.2 was normalized by the maximum of lengths of the two compared strings (manually extracted key phrase and the output of each method).

4.2.3 Evaluation of measured results

Unmodified

Data for this method shows, that error messages do not need to be always processed and sometimes the whole error message can be close to the key phrase.

MatchTemplate

As expected, this method returns the best results when a template exists for given error message. Since it is impossible to create a template for all types of error messages, only the most common should be considered to be implemented.

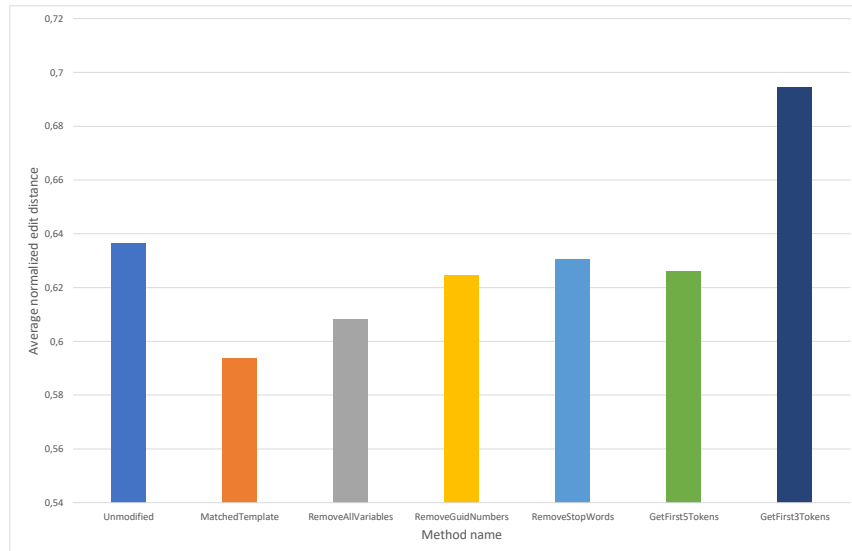


Figure 4.2: Average normalized edit distance from manually extracted key phrase for each method

RemoveAllVariables, RemoveGuidNumbers and RemoveStopWords

When implementing these methods for removal of dynamic parts of error messages and removal of stop words the amount of dynamic parts was overestimated. Even though these methods give better results than no modification they are mostly useful in combination with other methods.

GetFirstNTokens

The extraction of first 3 tokens gives bad results. The reason is that the key phrases are usually longer even when they are not at the beginning of the error message. Extraction of first 5 tokens on the other hand might be useful.

4.3 Solution relevance

In this section the same evaluation dataset of 50 selected error messages as in section 4.2 was used. However, the evaluation was done from the user's point of view. User is usually not interested in the edit distance from ideal key phrase, but in the relevance of retrieved results. Results of this evaluation are displayed in Figure 4.3. The difference between the total and the

4. EVALUATION

relevant number of solutions found is the number of False positives (see subsection 1.1.3).

The application was able to find a relevant solution to an error for 84 % of queries. The method `MatchTemplate` only found a solution 5 times out of 50 error messages, but its *Precision* = 1. On the other hand, the method `GetFirst3Tokens` found a solution 42 times and only 22 of them were considered relevant. Therefore, its *Precision* = 0.53 for this method.

During this test, the correctness of assigned tag was tested as well. The predicted tag was correct for 92 % of messages.

4.4 Tagging effectiveness

Tagging was implemented to increase the precision of the system by limiting the results only to StackOverflow questions that have assigned the same tag as the predicted tag. To evaluate the increase of precision, unmodified messages were sent directly to the StackExchange API. For untagged queries *Precision* = 0.77. For tagged queries *Precision* = 0.89.

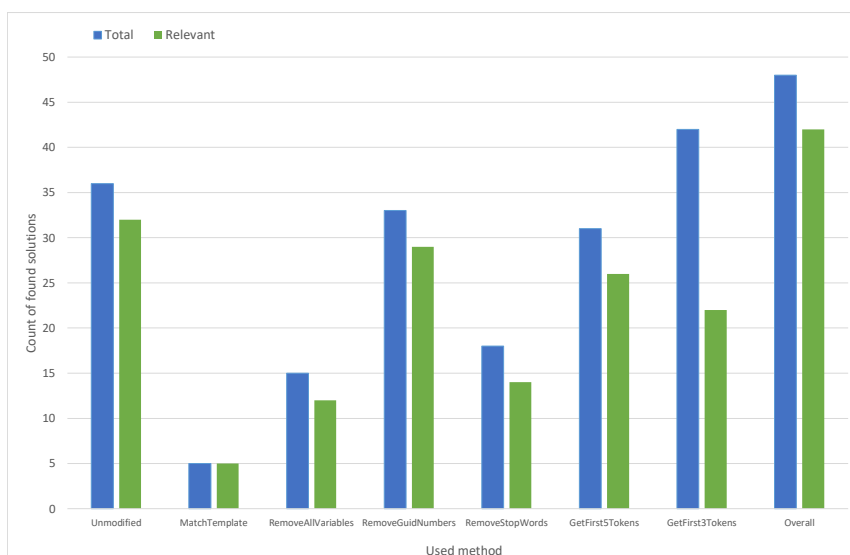


Figure 4.3: Count of total and relevant solutions found for each method

4.5 Performance

The average time for the web application to return retrieved results measured on 50 different queries is 5.30 s. This relatively long response time is caused by long response times of the StackExchange API. The calls to this API could be parallelized. However, as mentioned in section 3.2 the API allows one application to send only 30 requests per second. Therefore another measures preventing this limit to be exceeded would have to be implemented.

4.6 User testing

The target group for this application are programmers, developers or system administrators, who are responsible for solving errors in various applications. This application was tested by three users. Each of them was instructed to get solution for 10 error messages. They were also instructed to evaluate if any of retrieved solutions satisfied their information need and if the assigned tag was correct.

Selected users were programmers who each work with different technologies. They were briefly introduced to the application and since the functionality is very simple they started using it right away. All users understood how the application works and did not need any assistance when finding solutions to their error messages.

Database administrator

Works with SQL and Microsoft SQL Server.

- **Correct tag assigned:** 10/10
- **Solution satisfied information need:** 5/10

Business intelligence developer

Works with SQL Server Integration Services (SSIS)¹⁸.

- **Correct tag assigned:** 0/10
- **Solution satisfied information need:** 0/10

Web application programmer

Works with C# and JavaScript languages to create web applications.

- **Correct tag assigned:** 10/10

¹⁸<https://docs.microsoft.com/en-us/sql/integration-services/sql-server-integration-services?view=sql-server-ver15>

- **Solution satisfied information need:** 8/10

4.6.1 Evaluation of user testing

Overall, the tag was predicted correctly in 67 % of queries and one of retrieved solutions satisfied user's information need in 43 % of queries.

The application performed the worst for SSIS error messages. This has multiple reasons. SSIS is not in the list of selected tags, therefore the tagging module was not trained to recognize this type of error messages. Moreover since SSIS is not very commonly used on StackOverflow (not even in the first hundred of most used tags) it does not have many questions asked and answered. Last reason is that SSIS messages are unusually large and its harder to extract the correct key phrase.

On the other hand, it performed well for *C#*, JavaScript and even SQL errors. Retrieved solution satisfied user's information need in 65 % in these categories.

Conclusion

In this bachelor thesis we have analyzed current methods for key phrases extraction from text. Based on this analysis various key phrases extraction methods were implemented and used to extract key phrases from error messages. Even though statistical measures like TF-IDF or measures based on identifying uni-, bi- or tri-grams are commonly used for this task, the best approach was to use predefined templates. This is caused by the nature of error messages, which are usually short, contain many dynamic parts and are relatively structured.

The thesis also analyzed systems that provide solutions to error messages. StackOverflow was selected as the best source system for this thesis. It provides all the information needed, has big and active community, provides well documented API and is completely free to use.

Finally, we have analyzed information retrieval systems. This was needed to understand how they work to be able to use them as source of information. As a result of this analysis we have implemented own information retrieval system. This implemented system is focused on finding solution to an error using the error message.

To make the searching for solution as precise as possible we have implemented machine learning model, that predicts tag for error message. This model works very well for defined list of tags and has good evaluation metrics.

Although the application has interesting results, in the end of the process there must always be a professional who decides if the proposed solution is relevant and if it could be applied to the specific situation.

Future improvements

Even though the tagging module works well, and the created machine learning model has good metrics, it could be improved. This could be done by increasing the size of training data for example by integrating another system that contains tagged error messages.

CONCLUSION

Using this module, a ticket classification application could be implemented to automatically create bug reports and assign specific tag.

The key phrases extraction could be improved as well by implementing more templates to extract the key phrase. Another way of improving this module would be to implement an unsupervised machine learning model that would use some statistical measures to select the key phrase.

Bibliography

- [1] MANNING, Christopher D., Prabhakar RAGHAVAN a Hinrich SCHÜTZE. *Introduction to information retrieval*. New York: Cambridge University Press, 2008. ISBN 05-218-6571-9.
- [2] SKOPAL, Tomáš. *Boolean model of information retrieval: 2. lecture* [online]. [cit. 2020-05-30]. Available from: https://moodle-vyuka.cvut.cz/pluginfile.php/213162/course/section/33535/BIVWM_lecture02.pdf
- [3] Yang, F. , Dong, Z. and Liu, L. (2017) *Error Searching System with Keyword Extraction and Keyword Fuzzy Matching*. *International Journal of Communications, Network and System Sciences*, 10, 219-226. doi: 10.4236/ijcns.2017.105B022.
- [4] SPINK, Amanda a Bernard J. JANSEN. *Web search: public searching of the Web*. Boston: Kluwer Academic Publishers, 2004. ISBN 978-1-4020-2268-5.
- [5] *Search engine market share worldwide 2019* [online]. [cit. 2020-04-19]. Available from: <https://www.statista.com/statistics/216573/worldwide-market-share-of-search-engines/>
- [6] Kim, Yong-Mi. (2009). *The role of tags in information retrieval interaction*. *Proceedings of the American Society for Information Science and Technology*. 45. 10.1002/meet.2008.14504503137.
- [7] *About - Stack Exchange* [online]. [cit. 2019-04-29]. Available from: <https://stackoverflow.com/about>
- [8] NILLSON, Nils J. *Introduction to Machine Learning: An Early Draft of a Proposed Textbook* [online]. Stanford, CA 94305: Stanford University, Department of Computer Science, 1998 [cit. 2020-05-10]. Available from: <https://ai.stanford.edu/~nilsson/MLBOOK.pdf>

- [9] *How to build a dataset for your machine learning project* [online]. [cit. 2020-05-19]. Available from: <https://towardsdatascience.com/how-to-build-a-data-set-for-your-machine-learning-project-5b3b871881ac>
- [10] *Machine learning tasks -ML.NET — Microsoft Docs* [online]. [cit. 2020-05-21]. Available from: <https://docs.microsoft.com/en-us/dotnet/machine-learning/resources/tasks>
- [11] *Supervised vs. Unsupervised Learning - Towards Data Science* [online]. [cit. 2020-05-30]. Available from: <https://towardsdatascience.com/supervised-vs-unsupervised-learning-14f68e32ea8d>
- [12] *HULTH, Anette. Combining machine learning and natural language processing for automatic keyword extraction.* Kista: Dep. of Computer and System Sciences, Stockholm University, 2004. ISBN 91-7265-894-0
- [13] *.NET machine learning* [online]. [cit. 2020-03-23]. Available from: <https://dotnet.microsoft.com/apps/machinelearning-ai>
- [14] *What is ML.NET and how does it work? - ML.NET — Microsoft Docs* [online]. [cit. 2020-05-30]. Available from: <https://docs.microsoft.com/en-us/dotnet/machine-learning/how-does-mldotnet-work>
- [15] *How to choose an ML.NET algorithm* [online]. [cit. 2020-05-11]. Available from: <https://docs.microsoft.com/en-us/dotnet/machine-learning/how-to-choose-an-ml-net-algorithm>
- [16] *ML.NET metrics - ML.NET — Microsoft Docs* [online]. [cit. 2020-06-01]. Available from: <https://docs.microsoft.com/en-us/dotnet/machine-learning/resources/metrics>
- [17] *Train a machine learning model using cross validation - ML.NET — Microsoft Docs* [online]. [cit. 2020-06-01]. Available from: <https://docs.microsoft.com/en-us/dotnet/machine-learning/how-to-guides/train-machine-learning-model-cross-validation-ml-net>
- [18] *.NET Framework Regular Expressions* [online]. [cit. 2019-04-29]. Available from: <https://docs.microsoft.com/en-us/dotnet/standard/base-types/regular-expressions>
- [19] *Overview of Visual Studio* [online]. [cit. 2020-05-05]. Available from: <https://docs.microsoft.com/en-us/visualstudio/get-started/visual-studio-ide?view=vs-2019>
- [20] *Stack Exchange API* [online]. [cit. 2020-03-23]. Available from: <https://api.stackexchange.com/docs>

- [21] *C# - Error - Unable to access the IIS metabase - Stack Overflow* [online]. [cit. 2020-05-30]. Available from: <https://stackoverflow.com/questions/12859891/error-unable-to-access-the-iis-metabase>
- [22] *Use AutoML API* [online]. [cit. 2020-03-23]. Available from: <https://docs.microsoft.com/en-us/dotnet/machine-learning/how-to-guides/how-to-use-the-automl-api>
- [23] *What is Azure?* [online]. [cit. 2019-04-28]. Available from: <https://azure.microsoft.com/en-us/overview/what-is-azure/>
- [24] *What is .NET?* [online]. [cit. 2019-04-28]. Available from: <https://dotnet.microsoft.com/learn/dotnet/what-is-dotnet>

Used shortcuts

- GUI** Graphical user interface
- NLP** Natural language processing
- API** Application programming interface
- MSDN** Microsoft developer network
- JSON** Javascript object notation
- ML** Machine learning
- AI** Artificial intelligence
- IR** Information retrieval
- Tf** Term frequency
- Idf** Inverse document frequency
- CSV** Comma separated values
- REST** Representational state transfer
- SDK** Software development kit

Contents of attached medium

Visualise the contents of enclosed media. Use of `dirtree` is recommended. Note that directories `src` and `text` with appropriate contents are mandatory.

```
├── readme.txt ..... the file with medium contents description
├── src ..... the directory of source codes
│   ├── SolutionFinder ..... the directory of SolutionFinder solution
│   ├── thesis ..... the directory of LATEX source codes of the thesis
│   │   ├── img ..... the thesis figures directory
│   │   ├── *.tex ..... the LATEX source code file of the thesis
│   │   ├── FIThesis.csl ..... the LATEX class file of the thesis
│   │   └── zadani.pdf ..... assignment of the thesis
├── text ..... the thesis text directory
└── thesis.pdf ..... the thesis in PDF format
```