Comenius University, Bratislava
Faculty of Mathematics, Physics and Informatics

Semantic matcher for Obziva.sk

Master Thesis

2016
Adam Bilisics

Comenius University, Bratislava
Faculty of Mathematics, Physics and Informatics

Semantic matcher for Obziva.sk

Master Thesis

Study programme:	Cognitive Science
Study field:	9.2.11. Cognitive Science, applied informatics
Department:	Department of Computer Science
Supervisor:	RNDr. Kristína Malinovská, PhD.

Bratislava, 2016
Adam Bilisics

43310566

Comenius University in Bratislava
Faculty of Mathematics, Physics and Informatics

THESIS ASSIGNMENT

Name and Surname:	Bc. Adam Bilisics
Study programme:	Cognitive Science (Single degree study, master II. deg., full time form)
Field of Study:	Cognitive Science
Type of Thesis:	Diploma Thesis
Language of Thesis:	English
Secondary language:	Slovak

Title: Intelligent semantic matcher for Obziva.sk

Aim: 1. Create an intelligent semantic matcher for the portal Obživa.sk, which will automatically analyze verbally specified demand and find offers that most closely match the desired service.
2. For the matcher design, study, evaluate, and select the appropriate machine learning methods, focusing on methods inspired by cognitive processes.
3. The system should also learn from its users, who will indicate whether their search results are relevant to their query.

Literature: Allemang, D. and Hendler, J., 2011. Semantic web for the working ontologist: effective modeling in RDFS and OWL. Elsevier.
Kleedorfer, F., Busch, C.M., Pichler, C. and Huemer, C., 2014. The Case for the Web of Needs. In IEEE 16th Conference on Business Informatics, 1, 94-101.
Shvaiko, P. and Euzenat, J., 2005. A survey of schema-based matching approaches. In Journal on Data Semantics IV (pp. 146-171), Springer.

Annotation: The problem of intelligent data search on web portals poses a challenge for cognitive science inspired machine learning methods based on semantic relationships between words. Obziva.sk is a service offer-demand portal, which contains an interestingly large set of live, real-world data as well as maintains a busy user traffic allowing successful experimentation with its users.

Keywords: semantics, web portal, search, semantic matching

Supervisor:	RNDr. Kristína Malinovská, PhD.
Department:	FMFI.KAI - Department of Applied Informatics
Head of department:	prof. Ing. Igor Farkaš, Dr.
Assigned:	15.10.2014
Approved:	15.12.2015

prof. Ing. Igor Farkaš, Dr.
Guarantor of Study Programme

.... Student

.... Supervisor

43310566

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta:	Bc. Adam Bilisics
Študijný program:	kognitívna veda (Jednoodborové štúdium, magisterský II. st., denná forma)
Študijný odbor:	kognitívna veda
Typ záverečnej práce:	diplomová
Jazyk záverečnej práce:	anglický
Sekundárny jazyk:	slovenský

Názov: Intelligent semantic matcher for Obziva.sk
Inteligentný semantický párovač pre Obziva.sk

Cieľ: 1. Vytvorte inteligentný sémantický matcher (párovač) pre portál Obživa.sk, ktorý bude automaticky analyzovať verbálne špecifikovaný dopyt a nájde ponuky, ktoré sa najviac približujú požadovanej službe.
2. Za účelom dizajnu systému, preštudujte, vyberte a vyhodnoťte vhodné metódy strojového učenia s dôrazom na metódy inšpirované kognitívnymi procesmi.
3. Navrhovaný systém by mal mať možnosť učiť sa od užívateľov portálu, ktorí ohodnotia relevantnosť vyhľadanej ponuky k ich dopytu.

Literatúra: Allemang, D. and Hendler, J., 2011. Semantic web for the working ontologist: effective modeling in RDFS and OWL. Elsevier.
Kleedorfer, F., Busch, C.M., Pichler, C. and Huemer, C., 2014. The Case for the Web of Needs. In IEEE 16th Conference on Business Informatics, 1, 94-101.
Shvaiko, P. and Euzenat, J., 2005. A survey of schema-based matching approaches. In Journal on Data Semantics IV (pp. 146-171), Springer.

Anotácia: Problém inteligentného vyhľadávania dát na webových portáloch je výzvou pre kognitívne inšpirované metódy strojového učenia využívajúce sémantické vzťahy medzi slovami. Obživa.sk je portál pre ponuku a dopyt služieb, ktorý obsahuje veľký súbor živých, reálnych dát a má rušnú prevádzku umožňujúcu experimentovanie s používateľmi portálu.

Kľúčové slová: sémantika, webový portál, vyhľadávanie, sémantické párovanie

Vedúci:	RNDr. Kristína Malinovská, PhD.
Katedra:	FMFI.KAI - Katedra aplikovanej informatiky
Vedúci katedry:	prof. Ing. Igor Farkaš, Dr.
Dátum zadania:	15.10.2014
Dátum schválenia:	15.12.2015

prof. Ing. Igor Farkaš, Dr.
garant študijného programu

[illegible]

študent .... vedúci práce

Acknowledgement

I would like to thank my supervisor RNDr. Kristína Malinovská, PhD. for her help, guidance and advice. My beloved parents for giving me the opportunity to study on this faculty and their support. Last, but not least, I want to also thank my fiancée Kristína Slováková for all her love, patience and support during my studies.

Abstract

The problem of intelligent data search on web portals poses a challenge for cognitive science inspired machine learning methods based on semantic relationships between words. Obziva.sk is a service offer-demand portal, which contains an interestingly large set of live, real-world data as well as maintains a busy user traffic allowing successful experimentation with its users.

In this paper we present our own solution to the search problem by creating intelligent semantic matcher that provides matching offers for the users' demands, using the most recent methodologies such as machine learning and semantics.

After creating the network of words' relationships by the supervised learning function, the results of testing showed that our matcher is capable of providing users the correct search results with 47% accuracy.

Keywords: semantics, semantic matcher, supervised learning, web portal, search

Abstrakt

Problém inteligentného vyhľadávania dát na webových portáloch je výzvou pre kognitívne inšpirované metódy strojového učenia využívajúce sémantické vzťahy medzi slovami. Obživa.sk je portál pre ponuku a dopyt služieb, ktorý obsahuje veľký súbor živých, reálnych dát a má rušnú prevádzku umožňujúcu experimentovanie s používateľmi portálu.

V tejto práci prezentujeme naše vlastné riešenie problému vyhľadávania, vytvorením nášho vlastného inteligentného sémantického párovača, ktorý poskytuje zodpovedajúce ponuky pre dopyt užívateľov, s využitím najnovších metód strojového učenia a sémantiky.

Po vytvorení siete vzťahov medzi slovami funkciou učenia s učiteľom, výsledky testovania ukázali, že náš sémantický párovač je schopný poskytnúť užívateľom správne výsledky vyhľadávania s 47% presnosťou.

Kľúčové slová: sémantika, sémantický párovač, učenie s učiteľom, webový portál, vyhľádávanie

Contents

List of Figures

Chapter 1

Introduction

Computers and smartphones are inseparable part of our lives, same as all technologies that come with them. We could not imagine a day without using Facebook or Google, web applications that process great amount of data in order to function. For them to offer us requested and targeted content, they have to work with these data the right way. Everyone knows that Facebook does not show us content based just on our preferences, but also based on analyses of our posts or chat. Same applies for Google, which gathers any information and tries to offer us the best search results. This process is called data mining, where application is trying to mine as much information from the document as possible.

Depending on application each needs different information. For example Google wants to understand not only what user is looking for, but also content of “all” the existing websites to know what to offer him. At first this problem was solved by meta-data where developers had to fill keywords by which they wanted to be found. These words had to be verified so the algorithm also scanned the website, checking whether these keywords are even used in the content. This was also abused and web-masters just wrote these words anywhere without any meaning. To outsmart them search engines created algorithms that are trying to really understand the context in which the word is written.

This is where semantics, the study of meaning, became very important and each of these companies invested lot of capital into development and enhancement of semantic algorithms resulting into semantic search, which they all have partially implemented.

The main focus of this thesis is to create semantic matcher, which is, compared to semantic search, not just using semantic techniques, but also creates graph-like structures stored in the database. Matcher can then use these, to create semantic bonds between them and reflect real world understanding and relationships which would not occur in regular search. Not every semantic search engine knows that there is correlation between “color red” and “anger”, but well trained semantic matcher should be able to create and reproduce this bond.

Semantic matcher does not take each word separately, because we have to realize that if two words have the same “root word”, they have the same meaning, taking into account there is no negation prefix. Some kind of word “normalization” takes place and it is called stemming, where the input is requested word and output is it is root. More about semantics and search engines is explained in the subchapters number 2.1 and 2.2.

After this we examine existing stemming techniques in subchapter three, comparing most popular and well known stemmers like Lovins's stemmer [8], which was the first stemmer published, containing great number of rules. Later Mr. Martin Porter concluded his own stemmer from Lovins's work, being a big step in semantics and even nowadays there are variations of his algorithm for different languages.

Of course nowadays online search engines do not rely just on these techniques, but also on willingness of developers to hand over more structured data. This is possible by implementing various data formats into websites. We discuss online data formats in subchapter 5 and how they came a long way since their beginning.

Explaining different semantic search methodologies, we discuss these structures for storage of semantic data and explain why they are better for semantic search than ordinary database structures. We also considered different search techniques introduced in chapter 3, to describe dilemma why we will not use regular approach of full text search and its disadvantages.

Semantic search that does not learn over the time is useless. To make it “intelligent” it is possible to use different machine learning techniques which are either supervised, unsupervised or based on reinforcement explained more deeply in chapter 4. Based upon our decision to go with graph patterns semantic search methodology, which requires weighted graph implementation, we have to create paths or edges between words and assign them their weights. Changing these weights, potentially resulting into improvement of future results. In our methodology to improve results we take into account user inputs, even passive. That is why the only option for us is supervised learning where the user plays the role of teacher and our algorithm will be the “machine”, that learns from them and changes relationships based upon whether the user got the correct or incorrect result for his query.

An important question in semantic search and matching is the problem of how to distinguish relevance of words and make the machine/system prefer semantically more valuable choices. Answer can be found in chapter 5, where we present word evaluating algorithms. These are able to assign values to words and estimate which words are important in the documents and which are not.

After providing a decent technical overview, in chapter six we present portal Obziva.sk, explain what problems we came across and why the implementation of an intelligent semantic matcher is important. Next chapter presents final product of this thesis, our matcher and its implementation.

Penultimate chapter describes three different methods of testing, used to evaluate matcher's capabilities. After the evaluation of the matcher we present results and the thesis is ended by chapter where we discuss results, and suggest possible enhancements of the stemmer for the future.

Chapter 2

Semantics and semantic search

2.1 What is semantics

Smile. Heartbreak. Everyone loves cookies. Each word, phrase or sentence has its meaning. When we hear a word “smile”, we immediately think of someone smiling and it makes us feel good. But that is not something machines can understand. Or can they?

Of course computers are not able to feel joy or sorrow, at least for now, but they can have representation of relationships between words, learn and eventually also connect smile with joy or good feeling. How to do that is trying to explain study of linguistic semantics,to figure out how to capture meaning of words. To do so, linguistics distinguishes different kinds of relationships between words. There are synonyms, where two different words have same meaning, like “buy” and “purchase" or on the other hand homonyms, when words do sound the same but have different meanings. Word “left” is homonym, because it can mean opposite to right or past tense of leaving some place.

In English same as in Slovak there are other relationships between words(antonyms, etc.), but we cannot omit other relationships. According to the book Understanding semantics[1], descriptive meaning refers to certain category. Let us take for example sentence “I don’t have a dog.”. The “dog” is a category, where dachshund or poodle belong and these are called denotations. They do not have to be necessarily just kinds of dogs. It can be your friend who just got a number of a pretty girl, when you say to him “You are the dog!”. This can make sense in English but we would never say “Ty si pes!” in this context. That is why we need to look at each language separately, since they have different syntaxes, grammar and specific phrases.

To make a computer program understand these relationships, we need to help it learn, which nowadays can be done many different ways. We can hardcode each option, which is inefficient, cognitively implausible and not sustainable. It is also possible to use statistical methods, which require a vast amount of data to learn. But in our case we use supervised learning technique, when we use as an input for matcher training data and evaluate the relationship between match and query(positive or negative). Over time matching gets better, but more about that in chapter 5.

Other kinds of relationships between words are grammatical constructions such as suffixes and prefixes. Adding one or more characters at the end of verb to fit referred subject/s is called grammatical conjugation. Suffix can also change the referred time, like “I’m ironing my t-shirt.” and ”The t-shirt was ironed.” or in Slovak “Ja žehlím tričko.” and “Žehlil som tričko.”. On the other hand prefix “a-” can cause negate word “symmetric” to “asymmetric”, same as in Slovak with “ne-”, where “nemám” is negation of “mám”. What we have to keep in mind is, that some prefixes and suffixes change the meaning of words, but others do not can be removed without loosing semantic information. This cognitive based process is called stemming and removing unnecessary characters returns roots of words.

2.2 Search engines & Data mining

In the book Semantic Web for the Working Ontologist [2], Dean Allemang and James Hendler emphasize the so called AAA slogan that accurately describes World Wide Web. AAA means "Anyone can say Anything about Any topic". Thanks to the openness of the Internet it is full of information, created by large scale of people. At the beginning of WWW only developers were able to offer information and their ideas to users, but it changed really quickly and now anyone can create content on web.

When first search engines were created it was natural for them to try to offer best search results to users and the only way to do that sustainably was to scan the content of websites and analyze it. These analyses were simple but got more complex over time and better in processing and extraction of information. This is called text mining or text data mining, because of the effort to mine the most out of the text and transform it into usable data.

As the book Mining Text Data [3] describes, one of the early techniques used to process text is called the Bag of words. It is just simple division of paragraphs and phrases into words which create this “bag”. Disadvantage of this approach is that “bag” lacks any context, grammar or even word order. The only thing recorded is the number of occurrences of the word which is used to evaluate its importance in the document. Eventually we are able to create any types of “bags“ which can represent categories like important words, clutches, prepositions or even “spam bag” used to separate important emails from spam.

To improve this simple process we have to realize that words presented in World Wild Web have different variations after declension, conjugation or adding earlier mentioned prefixes and suffixes. Logically plenty of these variations express the same word or have the same root of word without changing meaning the of the word. For this, search engines have to detect the language in which the text is written and then select the correct program to separate the word roots - the stemmer.

2.3 Stemmer

When we are trying to algorithmically analyze a text document, we need to realize what kind of data we want to get, so we decrease potential of loosing important information from text. To reach the goal of our thesis, we need to retrieve word occurrences in the documents for further usage. In this case we consider the words with same root of the word the same, that is why normalization of these words takes place. Linguistic algorithm called stemming is focusing exactly on extracting roots of words which, as presented earlier, differs by language, but the idea is more or less the same, i.e. to remove prefixes and suffixes of the word. A Study on Stemming Algorithms [5] gave us better understanding about dangers of stemming and overview of most popular stemming algorithms.

2.3.1 Problems with stemming

During stemming we can realise, that some words are so “deformed” that there is no telling what was the original word, or in better case they are just not usable. This can happen when over-stemming or under-stemming occurs [5].

Over-stemming

Over-stemming is worse since two words with different meaning can end up as the same stem(root word). This false positive case can happen when the root of the word begins or ends with predefined suffixes or prefixes, which are in the process removed and/or replaced by the wrong one. For these cases, stemmer has to have special set of rules and try to avoid over-stemming, usually by returning the original word. It is always a better choice than getting a word without a meaning or merging different words in one.

Under-stemming

On the other hand there is under-stemming, false negative error, when the word is not stemmed enough or not at all. In that case two words that should have fallen under the same root word have a different one.

Compromise

There is a thin line between over-stemming and under-stemming. If we want to have less over-stemming errors, we can get much more under-stemming cases and get even worse results and the other way around.

Language is too complex, hence there is no perfect stemmer available right now and there probably will not be in near future, although we will present different, commonly known, approaches that work really well.

2.3.2 Porter's Stemmer

In 1980 Mr. Martin Porter created stemmer [6], which quickly became standard for English language. As people unsuccessfully tried to create its variations with major flaws, Mr. Porter released programming language commonly known as Snowball [7]. This project allows developers to write their own stemming algorithm and it resulted into successful stemmers for other languages.

Porters stemmer is based upon 6 steps concluding 60 rules. Even though it is quick, limitations are obvious, since the prefixes have to be manually defined and algorithm does not apply its stemming function multiple times to one document.

2.3.3 The Lovins's stemming algorithm

First stemming algorithm, published by doctor of philosophy Julie Beth Lovins in 1968 [8], astonished public as it influenced the whole semantic area. It consisted of great number of suffixes (294 endings, which is much more than Porters).

The first step was to find the longest suffix. Second condition depends on in which category is the removed ending. Then algorithm chooses one of 29 conditions and applies it, for example if the root word is shorter than 3 go back to step 1 and find shorter suffix. Final step are transformation rules which also differ by the ending of original or already stemmed word. We can think of them as exceptions in endings. Limitations are similar as in Porter's stemmer defined by the number of endings.

2.3.4 Husk Stemmer

Difference between Husk's stemmer compared to Porter's or Lovins's is that it iterates. That means that after completing the first round of stemming, if any rule was applied, it stemms again until no suffix was found, or if the word's length is only 3 letters long and it starts with vowel, or finally if it is 4 characters long and starts with consonant. This is kind of rush approach that can result in many different words mapping to same root word.

2.3.5 Yet Another Suffix Stripper

Also called YASS is more convenient stemmer, since it is applicable to any language. Yes, there are variations of Lovins's or Porter's algorithms for many languages, but the fact that someone has to analyze the selected language and create specific rules is a big limit. YASS uses completely different approach where no linguistic input is considered.

At the beginning lots of documents is needed as presented in a paper by authors of this stemmer [9] to create lexicon of words. After this distances between words are calculated by comparing positions of characters where the words differ and using a specified formula. Then clustering by the distance takes place and divides words into homogeneous groups. In these groups it is easy defined what is the root of the word.

2.3.6 Which one is the best?

In the end, each approach has its own advantages and disadvantages, that is why we cannot say that one is better than all the others. But when searching for stemmer for specific language, all depends on if there is a specific stemmer technique already available or whether to use YASS as language independent stemming technique.

2.4 Semantics online

As we could see in the previous section, it is not easy to get desired data from the document. Even on the web the situation was not brighter at the begging. Provided structure of original HTML 1 - 4 had only low semantic value, providing crawlers just anchor titles. Titles did not have any attribute or structure to indicate whether it is a subtitle or title. Then developers were left with just paragraphs and styling elements, semantically useless structures, which were essentially raw content.

Over the years there were several attempts to get more structured data from the websites. These solutions had either embedded or external character.

2.4.1 Not embedded

One of the early approaches was to link HTML document with alternative data representation. Web syndication is an example of alternate representation of data that developers use even today. RSS channels are used to "broadcast" information from the site to any listener (another website or end user). RSS data are most commonly stored in XML format and the HTML document points to it:

```
<link rel="alternate" type="application/rss+xml"
```

```
href="http://example.com/feed" />
```

2.4.2 Microformat

Microformat [11] came also early with embedded solution and are dependent on the names of classes. This way they did not have to add any attribute to HTML, but still be able to semantically mark any element. Data can afterwards be transferred using JSON format, which is human-readable text format to transmit data objects such as arrays, and used by 3rd parties like search engines or crawlers. There are several editions and some websites are still using the first or the second edition and even today some microformats like hCard are used more often then other alternatives (Microdata or RDFa). Unfortunately the problem with microformats is usage of an attribute "class", for different purpose than it was supposed to be. That is where other solutions came.

Listing 2.1: Example of microformat h-card which represents contact information

```
<p class="h−card">
  <img class="u−photo" src=http://obziva.sk/photo.png" alt="" />
  <a class="p−name u−url" href=http://obziva.sk/AdamBilisics>Adam Bilisics</a>
  <a class="u−email" href=mailto:a.bilisics@gmail.com>a.bilisics@gmail.com</a>,
  <span class=p−street−address">23 Wall Street</span>
  <span class=p−locality">New York City</span>
  <span class="p−country−name">USA</span>
</p>
```

2.4.3 RDFa

Existing concept of W3C(World Wide Web Consortium) called RDF (Resource Description Framework) was already functional for XML. Its implementation in XHTML was proposed by Mark Birbeck to W3C in 2004 and later that year became part of next version of XHTML 2.0. Mark Birbeck with Steven Pemberton even managed to publish first official document about the syntax of Resource Description Framework in Attributes or shortly RDF/A [12]. RDFa used few of already implemented XHTML attributes like @rel or @content, but it also came with new ones @about, @property, @resource, @datatype and @typeof [13].

Since RDFa is variation of RDF, it also holds on to the concept of triples. Triple is the atomic data entity which says that any data can be stored in the form of three expressions: subject, predicate and object [2]. For example in the sentence "Adam has the best master thesis", the subject is "Adam", the predicate is "has" and the object is "the best master thesis".

Thanks to RDFa, semantic value of data is not lost in the process of displaying data from the database to XHTML. Even thought the popularity of RDFa at the time was high, it was too complicated to implement (see example below), so when Microdata came out in 2009, it took a fair share of the semantic market (almost half). The attempt to get back developers using Microdata, was a much simpler version of RDFa - RDFa light. This move was partially successful and made it easier for developers to implement it, but Microdata continued to gain followers.

Listing 2.2: Example of information about a person in RDFa

```
<div vocab="http://schema.org/" typeof="Person">
  <a property="image" href=http://obziva.sk/adam.png">
    <span property="name">Adam Bilisics</span></a>,
  <span property="jobTitle">Founder</span>
  <div>
    Phone: <span property="telephone">0903 444 444</span>
  </div>
  <div>
    E−mail: <a property="email" href="mailto:a.bilisics@gmail.com">a.bilisics@gmail.
        com</a>
  </div>
  <div>
    Links: <a property="url" href="http://obziva.sk/">Obziva.sk</a>
  </div>
</div>
```

2.5 HTML5 and semantics

To accomplish our goal to create the best semantic matcher, we new that all the gathered information should not be only for our purposes. That is why we looked through all the options how to make them available also for search engines and found the newest solution which goes hand in hand with HTML5.

2.5.1 WHATWG vs W3C

HTML 4 wasn not updated since 2000 and W3C was focusing more on XHTML. Lots of developers were not satisfied with this direction, so in 2004 they founded community called Web Hypertext Application Technology Working Group (WHATWG)[14], consisting of individuals from Mozilla, Opera and Apple. In 2007 WHATWG presented their proposal to

W3C and began collaboration on new standard HTML5 [15]. Unfortunately their opinions differed in whether to make HTML5 living standard or not and each organization went their separate way and created their own specifications of new HTML. WHATWG went with a living standard philosophy [16], where HTML is never finished and it is still improving (that is where they reduces HTML5 into just HTML), but W3C defined HTML5 as a new defined version and each version is specified by a number(currently HTML 5.1) [18].

2.5.2 What is new?

First of all, HTML5 made defining things like document type and charset much more easy.

Listing 2.3: HTML 4.01 document type declaration compared to HTML5

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
  http://www.w3.org/TR/html4/loose.dtd">

<!doctype html>
```

Listing 2.4: HTML 4.01 charset definition compared to HTML5

```
<meta http-equiv="Content-Type" content="text/html; charset=utf-8>

<meta charset="utf-8">
```

Until now the very popular, but old, technology Flash, was used by developers to play multimedia and even games on their websites. But HTML5 came with new solutions how to handle multimedia on the web with two new tags for video and audio formats which essentially finished existence of Flash and more importantly, it defined new pair tags to improve semantics on the web.

There was a long discussion about whether implement RDFa into HTML5 or not [19]. Long and not constructive discussion resolved into an article by Dr. Jenifer Fay Alys Tennison [20] where she summed up all arguments for and against this step, also expressing her own opinion.

Arguments of Tennison and other people from semantic community are straightforward. RDFa is complicated to implement and even usage of prefixes instead of URIs would not help much. Another complication is that all these data mean nothing without defined structure by an external URI. The word external is important in this context, since when the website(or server) goes down or the content of the URI will be removed, all data will lose meaning.

Because of all these disadvantages, the author of HTML5 specification Ian Hickson [21] proposed new concept of Microdata. This is where all the major names like Google, Microsoft, Yahoo and Yandex created collaboration known as Schema.org[22]. As it is in the best interests of these services, to get the best out of data on web, they defined vocabulary that can be used with different encodings, preferring Microdata.

2.5.3 Microdata

Philosophy of Microdata is based upon an idea that everything is an item. It doe not matter if we talk about person or a movie, the same attribute structure applies on every item on website. Item is specified by an attribute “itemscope” without any value and the begging and end of an item is defined by the pair tag containing this attribute, usually it is div .

After the search engine identifies an item, developer specifies what type of an item it is. Itemtype is defined in the same tag that itemscope is and can have same kinds of values as RDFa Lite, differing from event, organization or person, but they can also have any kind of custom type. As long as anyone defines it, it can be used by refereeing with URI. Even though itemtype provides context information, it is not required and search engines will be still able to identify properties of the item.

Itemprop attribute can be placed in any start tag which has also pair tag that ends it(pair tag), handing over specific information about the content inside the tag. As an example itemtype Movie can contain properties like actor, director, of course name and many more [23].

Another feature of Microdata is that we can define a new item inside another one, exactly the same way as we define regular item. For better understanding see following listing:

Listing 2.5: Microdata example of itemtype Movie structure from www.schema.org with another item inside

```
<div itemscope itemtype ="http://schema.org/Movie">
  <h1 itemprop="name">Avatar</h1>
  <div itemprop="director" itemscope itemtype="http://schema.org/Person">
  Director: <span itemprop="name">James Cameron</span> (born <span itemprop="
      birthDate">August 16, 1954</span>)
  </div>
  <span itemprop="genre">Science fiction</span>
  <a href="../movies/avatar−theatrical−trailer.html" itemprop="trailer">Trailer</a>
</div>
```

2.5.4 Why should we care?

The same question was posed when developers were proposed to use keywords. No-one will spend their time implementing Microdata into the website, unless it will bring some benefit.

Answer is simple: “Because search engines will like you more”. Not in a verbatim kind of way, but the website will gain points in search order and that is what everyone is seeking for. Everyone wants to be the first. If this is not enough, Google rewards each “semantic” website by displaying all the additional information in search results (see figure n. 2.1), making it more attractive for users.

Figure 2.1: Example from the book HTML5 & CSS3 for the real world [15] of how google displays additional information based on Microdata.

2.6 Semantic search methodologies

The world of web as we know it consists of many documents linked together. These links represent relationships between documents, and each of these relationships can be easily understood by humans. So the first question is, why humans? Because when reading website, we understand its context. As an example, when reading something about HTML5 on www.w3c.org [17] and there is mentioned Ian Hickson as an author of HTML5s specification. His name can be a link to a wikipedia article about him, which makes a prefect sense if we want to learn something about him.

Now from the perspective of search engine, there is a w3c subpage with title HTML5, referring to the wikipedia article with title Ian Hickson. What des mean HTML5 or Ian Hickson for search engine? Nothing special, just two linked websites, which for all that they know, can make no sense. But Ian is a person and HTML5 is a markup language, two different entities.

2.6.1 RDF Path Traversal [34]

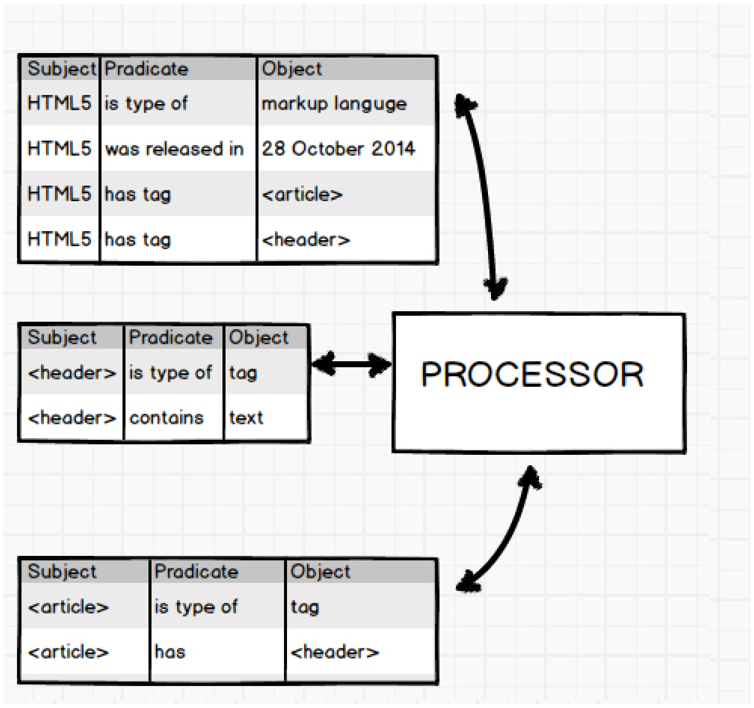
Semantics differentiates between words or sentences and refers to these entities as resources. A resource represents anything that anyone can talk or read about on the web. That is why it is also used in the name of RDF, which is Resource Description Framework.

RDF structure

As mentioned earlier in section 2.4, if we want to share the data through the Internet without any lost of meaning, it would have to be structured. Since sharing databases would not be secure, we have to share the data using the same structure. Solution to this data distribution problem was the RDF proposed by W3C in 1999. Usual structure in database is the table with any number of columns. On the other hand, RDF is based on a structure called tripple, consisting of subject, predicate and object. For the usual sql developers subject can identify a row of table, predicate a column and object is a value of the cell. By tripples we can capture essential information for preservation of meaning. Example of RDF can be found earlier in the section about RDFa or in Figure 2.2.

Distributed data

There are lot of websites using RDF or RDFa for data representation. Now when we have all these information in the right format, we need something to join them. Processor makes this possible, by accessing all these data and connecting it in the form of graph(see Figure 2.3).



Subject	Pradicate	Object
HTML5	is type of	markup languge
HTML5	was released in	28 October 2014
HTML5	has tag	<article>
HTML5	has tag	<header>

Subject	Pradicate	Object
<header>	is type of	tag
<header>	contains	text

Subject	Pradicate	Object
<article>	is type of	tag
<article>	has	<header>

Figure 2.2: Example simplified RDF data and processor connecting them

Traversal in RDF Graphs

Desired result for the semantic search would be to find the best match for our query by processing it and finding corresponding nodes in the RDF graph. Each node refers to all related matches for the query. For example: if someone searches for term HTML5, they get all the information about it and also related tags as in figure n. 2.2.

This semantic search is actually just traversing the graph created by processor. The first option is traversing the graph, by starting in one node and requesting all related instances. For this purpose there is SQL-like query language SPARQL , specially created by W3C in 2008[24] for RDF. For PHP developers the basic difference between SPARQL and MYSQL is that in SPARQL we have to define exact type of each attribute.

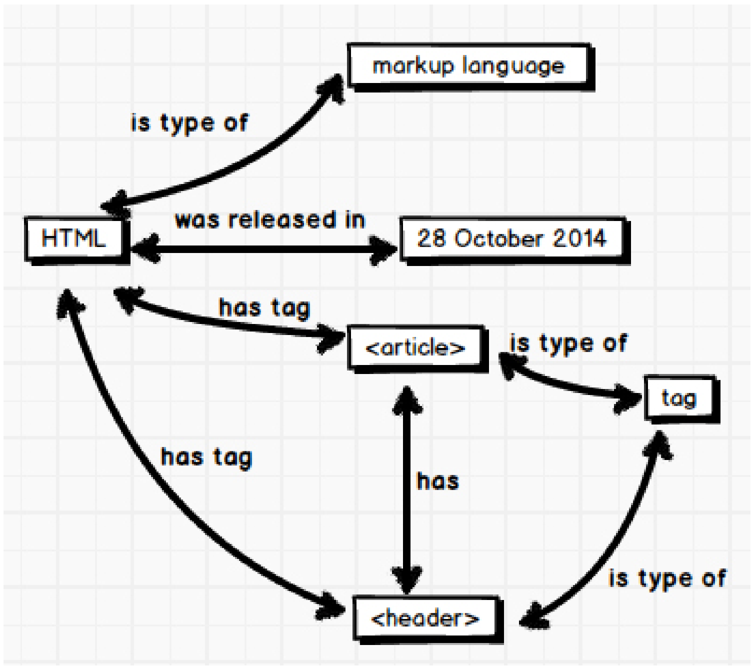


Figure 2.3: Combined graph of all tripples

Listing 2.6: Querying address of a person by name and surname in SPARQL. In this case

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?address
WHERE {
  ?person a foaf:Person.
  ?person foaf:name ?name.
  ?person foaf:surname ?surname.
}
```

The second approach are other query languages, which use different traversal techniques, where the ontological information about the domain is important. This means, that the search is happening only in the certain area of graph which corresponds with the domain and focusing on relationships between domains nodes.

2.6.2 Keyword to concept mapping

The easiest way for computers to understand information is to use usual knowledge modeling methodologies which are based on a non-ambiguous philosophy. This is useful, but non programmers will never know what it represents. On the other hand, concept maps are primarily focusing on cognitive aspect — how humans process language. Therefore, maps are trying to represent knowledge in the same form as humans do.

Concept maps are commonly used by designers, engineers or anyone else, who is focusing on a certain topic and the words or phrases related to it. These maps are visualized by graphs in the form tree where the root is the most general word and as we go down to the bottom we get more and more specific concepts. Language is really complicated and these maps are usually created by humans, but in case of semantic concept mapping used by search engines it has to be algorithm based. For transforming language data into an unambiguous form, we need to use structural analyzes and semantic analyzes.

The structural analysis of a sentence can be represented graphically, when the concepts are abstracted into articles and relationships by arcs or arrows, which is a graph structure like in Fig.2.4. We have to keep in mind that this type of analysis is strictly from a structural perspective. On the other hand semantic analysis is focusing on the part where the meaning needs to be captured.

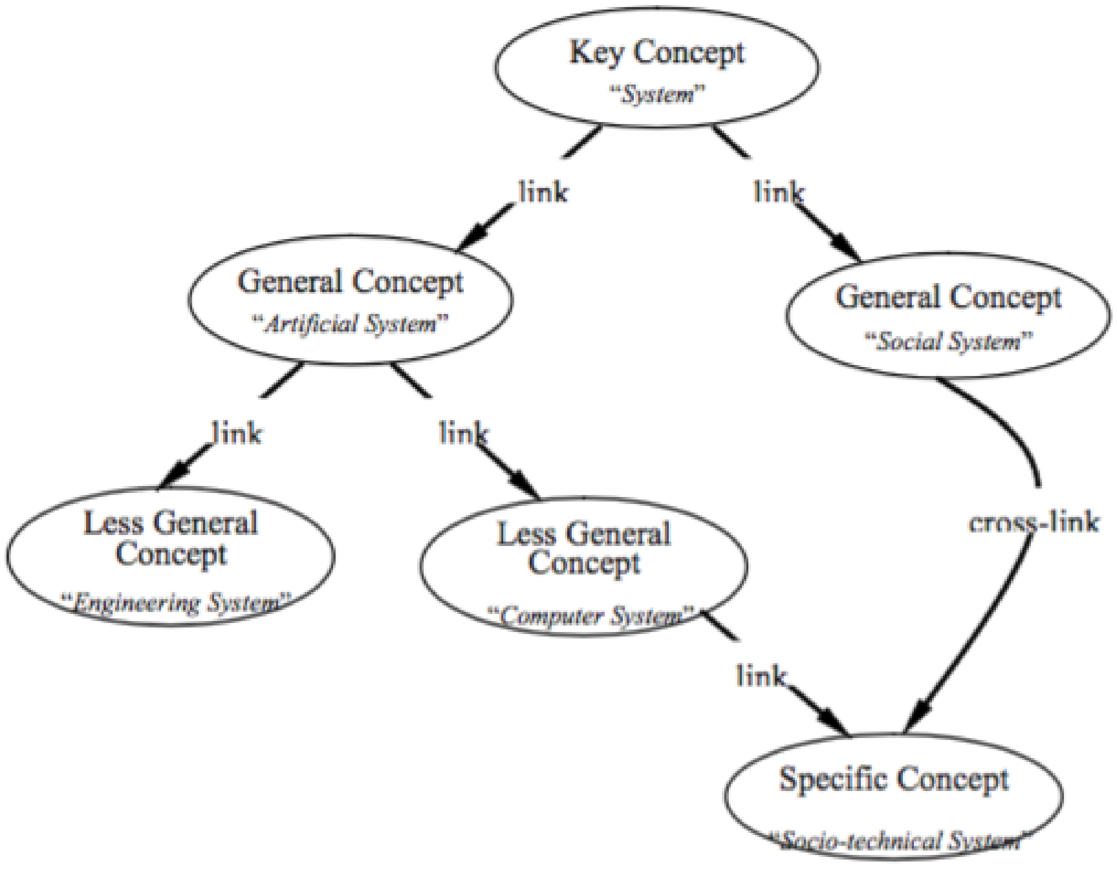


Figure 2.4: Concept mapping graph taken from [25]

Jin-Xing Hao, Angela Yan Yu and Ron Chi-Wai Kwok [25] present their own solution of map based knowledge modeling. From their point of view we first have to realize that it is easier to create links between single words, or accurately called labels, based on dictionaries or other data linking them, rather than complex expressions. In their research the links are not as important as the concepts and focusing on this fact they define the concept map as a formula:

$$S = (T_c, T_r, O)$$

T_c represents complete graph of labels and their semantic relations, T_r set of binary relations and O stands for ontology, which is a hierarchy of words in specific domain. To easily compute semantic relatedness of two words, they used WordNet[26], which is a lexical network, with all words organized into sub-networks of synonym sets. This solution was tested by Rubenstein and Goodenough's benchmark, commonly used to evaluate semantic measure and returned satisfying results.

2.6.3 Graph Patterns

Since RDF data have a graph structure, some searching algorithms are using a concept to pattern mapping, where the query is transformed into specific graph pattern. This pattern is created the same way, as the whole graph we are searching in. If this pattern or its subgraph is found, the result is displayed. Some approaches also preprocess the graph, while the patterns are extracted by an algorithm, enabling creation of relationships between patterns. We can image it is like finding a place for a puzzle piece. We do not have to try it everywhere, if we know its pattern and where in the puzzle it is missing we do not have to try to put it everywhere. Therefore we only have to try to place the piece into a smaller amount of places.

Chapter 3

Search options

Before implementing our cognitive solution of search, we had to consider other search methodologies that could possible work better for our needs.

3.1 Linear search

The linear search [28] is oldest and easiest of all since the algorithm has to look through all the rows in database and has the worst search time, which is linear - O(n). This means that if we have *n* rows it has to male *n* comparisons. One of the ways how to improve this search is knowing whether the requested value is closer to the beginning or the end of our database which is not applicable in our case. Another reason regular search would not work for us is that expected queries will be longer sentences that differ from all the offer descriptions and comparing just words would result in a lot of bad results.

3.2 Boolean search

Boolean search is more advanced as it allows anyone, not just developers, to create more complicated queries with boolean operators *AND* - conjunction and *OR* - disjunction. These queries result in two statements TRUE and FALSE depending on whether the result was found or not, meaning that search engines return results that are TRUE. Study conducted by Diane Nahl and Violet H. Harada [29] investigated capabilities of people without any programming skills, whether they are able to create queries after simple explanation of boolean search concept. Assignments were very simple as test subjects had to create queries with just two words. For example in one assignment they had to create query that returns terms “teenagers” and “gangs” from sentence “Why do teenagers join gangs?”. The correct answer is to request “teenagers OR gangs”. The final results indicate that only 60% of their queries were correct which means that if we would implement boolean search, the users could get upset or frustrated and leave out portal.

3.3 Tags

Tagging is phenomenon started by creators of HTML standard [17] implementing new meta-tag with attribute “keywords”. Developers were finally able to define keywords by which search engines evaluated search results for user’s queries. The same approach was applied by bloggers to tag their blog posts creating easier access for readers to their favorite topics.

Same behavior occurred on social networks like Twitter, Instagram or Facebook, when people started tagging their posts to express what their content is about and to gain interest of other users. As tagging gained on its popularity many complicated portals implemented this feature, since the usual categorization was not enough. In Slovakia portal SAShe.sk [27], where people sell their hand made products, implemented tagging and allowed users to influence their search. Unfortunately there is still a risk of people not searching by tags, which means it is not sufficient and usually developers implement also other search option for these cases.

3.4 Full-text search

Full-text search is much more intelligent than other search options where full-text means that the search query is not compared as is, but rather broken down into separate words. There are two options of how to implement full-text search: direct method and indexing.

3.4.1 Direct full-text search

In this method search engine compares each word from query with each row in database directly, counting its occurrences. The problem with direct searching emerges when the database has too many records, resulting into slow response. When the search engine is too slow, there is a high possibility of loosing users which is why the best way to implement full-text search for large databases is Indexing.

3.4.2 Indexing

To enhance search engine’s speed it is necessary to modify the database into full text database by adding new database table where all the words will be stored. After the database is ready, all of the documents are divided into separate words and each word is inserted into the words table along with index of the document where it was found. Searching process is now focused of founding the words from query in the words table counting all the occurrences of words in each offer as evaluation of probability that the result is correct and returning indexes of these documents.

3.4.3 Linguistic enhancement

To improve search results, it is possible to apply stemmer[30] before inserting words into database. This will save the searching time as the words with same root of the word will not have separate records in the database and the search engine will be more likely to return correct results for the stemmed query, since the user does not have to necessarily use the same form of the word as the desired document.

Chapter 4

False positive problem

All search engines have to face false positive problem when the algorithm displays bad search suggestions. There is no simple solution to this problem and it is necessary to use advanced methodologies like learning or determination of the words' importance.

4.1 Learning

When the first computers were build and we realized their potential, the new subfield of computer science emerged, trying to pass our knowledge onto them - Machine learning. To acquire knowledge means to learn, but the usual algorithms are explicitly programmed, which means that the program can do only what and how it was programmed to. Fortunately we are able to create learning algorithms which can learn 3 different ways [31].

4.1.1 Supervised learning

The most intuitive learning for humans is supervised learning, since we also learned the most of what we know from our teachers. Mom, dad or teacher supervised our activity and helped us to lean it better. The same applies for supervised learning of computer, where the teacher gives a computer algorithm examples and evaluates the output or provides correct output values upfront.

After the certain amount of learning data, the algorithm should be able to create a concept by which it will be able to evaluate almost any input correctly. It is also not unusual to design the application to keep learning, based on users input. This approach is wildly used by search engines as the search result position partially depends on users' visits.

4.1.2 Unsupervised learning

As the name of this method suggests, there is no teacher to tell the algorithm whether the output is correct or incorrect. Predefined rules can be applied under some conditions, but the final decision is always up the algorithm. It would be hard to create such model for searching as the goal depends on user's input and his decision to choose one of the search results.

4.1.3 Reinforcement learning

To understand reinforcement learning we have to comprehend what is software agent. Agents are entities in Machine learning placed into the environment. The environment can have a usually visualized digital form of a program or have a physical form where agents are represented as robots.

The reinforcement learning method is inspired by behavioral psychology as it is based on environment and agents. The behavior of agents is not controlled or influenced by supervisor by any way, instead they learn from their experience in the environment and reactions in form of inputs it provides for each action.

4.1.4 Search learning

To summarize all the learning methods, both unsupervised and reinforcement learning would be hard and impractical to teach search engine return better results, since there is no feedback for users to provide. On the other hand supervised learning can be easily used for this purpose as its learning is fundamentally based on inputs from teacher.

4.2 Importance of the words

The task of finding the right search results for the query is complicated as all the most of the information they work with is text based. Simply comparing the similar words of query and possible result does not say anything about the context or the importance it has. For this purpose search engines use many different information retrieval methods to gain these important data. We picked one method that is the most used, easy to implement and show the best results.

4.2.1 TF-IDF

Term frequency - inverse document frequency is a information retrieval method based on statistical analyses of the documents, evaluating each word by a number which reflects its importance in the document set. The set of documents, also called corpus contains all the documents of which words we need to evaluate. The basic formula for its computation

$$TF - IDF = o * log(\frac{D}{dt})$$

where o is the number of total occurrences of the word in all the documents, D is the number of documents and dt is number representing the subset of documents that contains this word.

The number produced by TF-IDF can variate from 0 to unlimited and for the further computational purposes it needs to be normalized. The logarithmic part which represents IDF, is already normalized by the logarithm. On the other hand term frequency is not and we can achieve normalization by applying logarithmic normalization:

$$TF = 1 + log(o)$$

After normalizing the term frequency the highest value depends on multiplication of both mentioned logarithms and their bases.

Chapter 5

What is Obziva.sk

5.1 Handy people of Slovakia

Each and every person is talented. Someone is good in math, informatics or physics. Other people are more handy in car repairs, fixing plumbing, or even doing usual home chores. It does not matter what it is, as long as we love our work. Unfortunately, a lot of people end up doing some other job because it pays better or their favorite activity is not work related. In Slovakia this was reality until few years back. That is when people discovered potential of entrepreneurship and started their own business. Of course, at the beginning they did not have much money, but they needed to acquire some customers. This is where the power of the internet came in equation, especially Facebook, where people created Facebook group called "Susedská obživa". Not only entrepreneurs, but also ordinary people without work that needed some money, offered theirs services by posting in the group repeatedly, until it became really annoying and upgrade was required. This is where www.obziva.sk was created, to bring all the offers together in categorized environment and most of all user friendly.

5.2 Basic users

From the beginning the portal was focused on usability and tried to offer the most intuitive structure. To avoid duplicity of offers, we decided that it would be the best co come up with categorization system, enabling users to select services from the predefined list and add the description to their offers.

Unfortunately, in Slovakia maternity leave is very low, and that is why after the portal was deployed a lot of users, especially mothers added almost every service from the list. Even ones that were very specialized like repairing computers or as ridiculous as moving someone to new apartment. Apart from this fact, the credibility of the portal was in stake, since almost everyone was in every category. To solve this problem we were forced to manually remove lot of offers and restrict the number of offered services to 5.

5.3 Categorization

Problem of categorization on Obziva.sk was also a big challenge, since we did not want people to add new categories by themselves. Our categorization of services has two levels. The first one are main categories and the second one are subcategories. Each category or subcategory also had to be linked with its own icon, that intuitively describes it. To solve this problem we studied other portals and their categorization, but there was always some category or subcategory missing. And we could not keep adding categories by ourselves, since we are not experts in every existing service field and additional subcategory in each category called “Other” was growing.

5.3.1 Cognitive approach

As part of our semestral project we aimed at searching for a cognitively inspired approach that could help us to solve this problem. As a result of our work we came up with idea to analyze already existing offers, and get the most relevant words extracted from the offer’s description as suggestions for new categories using a suitable intelligent search algorithm. In this project we taken into account other highly cognitive inspired solutions like L’udovít Malinovský in his paper [32] where he tried to solve the problem of categorization by applying distinguishing criteria, but for our purposes we chose more effective and populat TF-IDF solution.

5.3.2 Same words

First of all we needed to make sure we will not get duplicates in our result, but the but the word can be conjugated many ways and have suffix or prefix. Since we needed just a root of the word implementation of the stemmer was clear choice. That time we studied mainly Porter’s Snowball [7], and found its official implementation for Czech language which is really close to Slovak.

5.3.3 Tf-idf and categorization results

After extracting all the root words with Ljiljana Dolamic and Jacques Savoy’s stemmer [41] we saved each offer in stemmed form to be able to apply statistical evaluation method “term frequency - inverse document frequency” [33] described in subchapter 4.2.

Finally we analyzed 125 descriptions with highest TF-IDF value of 0.3. The highest rated words contained some of the already existing categories, but also suggested new ones like “lakovanie” or “pneuservis”. After partially resolving problem of categorization we did not found any other problems with offerings, but there was still an issue with demands.

5.4 Problem with demands

Ironically Facebook page, that was meant to be just for advertisement, became more used and more popular than the website. Even though, there is more then 700 offers of services on www.obziva.sk, people looking for these services do not want to go the website and search through the categories. They rather end up writing a post on Facebook page in hope, that someone will see it and write them a comment or message.

Based on this circumstance we decided to think “out of the box” and ask ourselves a simple question “How can we bring users on www.obziva.sk with such convenience of usage, as writing request in one field, like on Facebook?”. Answer was simple, to provide the users with the possibility to search (effectively).

Chapter 6

Inspiration - Web of Needs

World Wide Web started with HTML documents containing text information, that could be interesting for visitors. Later, developers of web standard and web browsers came up with various options how to display different kinds of colors, images and even videos. A whole new era came, opening an opportunity for regular people to express themselves. Now with minimum effort anyone can say anything about any topic (AAA slogan mentioned in subchapter 2.2) and if there is a demand for something, there is an offer. That is what the Internet became, a medium where we can exchange information in any kind of form. This offer/demand marketplace was also useful for sellers, that could start selling their products over the Internet.

Florian Kleedorfer, Christina Maria Busch and their team see the Internet through Web of Needs(WoN) [35] as the big marketplace, where there is always someone offering something, and always someone looking for it. And because of this view, they see also weakness of this marketplace and that is how to connect the right supply with the right demand. Semantic matcher WoN is focusing on e-commerce and believes that there is a better way to connect these two parties than what usual search engines offer, and that is to create an environment where the search understands the context and the sentence itself and it is not limited to searching by keywords. Place where people can also communicate about the product, or service. One simple way how to get what we need. This is the WoN author's own way how to look at the Internet. Everything is a need. Either way we need to offer something, or we need to find something, hence Web of Needs.

6.1 WoN philosophy

Before designing the architecture, the creators of WoN had to consider the whole philosophy behind it. In case of an e-shop, it is useful to know not only desired product that the user wants to buy, but also his age, gender, previous buys and other information, that would help to display also other custom products, interesting for this person. WoN does not need these information, since it is matching service working with many sites that offer products or services and the only goal is to give the user what he came to find. It is also not just focusing on the requestors side as in case of an e-shop, nor only the side of supplier. This matcher considers both parties as same, because they are both needs. Now the important question raises: "What kind of input do we want from the user (doesn't matter which side of transaction it is), when creating the need?".

6.1.1 Required input

When describing the service or the product, there is a required level of description, to be able to identify the desired need. Generally, there are three input fields used for this purpose: title, description, and tags. By entering the title we are able to describe in a short way whether it is a product or a service and to define its category. The purpose of description is to specify all attributes and details about "our need", like the brand of shoes, width and height of a surfing board or the location where we need a service to be executed. Tags are usually not a required field they only help to increase search accuracy, much like keywords.

6.1.2 Contacting the other party

After getting some matches for the need, requestors often have some questions about the service or just want to verify its quality. Nowadays we can find contact information within offers page, or in case of e-shops there could be live chat. Search engines don't have such options and users have to find them and make a contact with the service party. A matcher could solve this problem by connecting these two parties directly after the request for communication is accepted. In the WWW terminology this is called handshake.

6.1.3 Advantages

The presented logic of the matcher allows a service to contact the requestor, hence the requestor can get what he needs without any effort what so ever. He is no longer dependent on his searching skills or have to rely on recommendation systems. Requestor just posts his need and waits if there is any service that satisfies it.

We have to realize that this is a completely new approach, which changes the nature of services. Until now they were inactive entities, just offering their services or products on website and waiting for an email or a phone call. WoN allows services to take the initiative and contact all potential or even adjust offer by demand. Which results into better and healthier marketplace and also satisfies users with special requests that are not usually available because of low demand.

6.1.4 Problems

The scenario in which this kind of marketplace gets popular and connects lot of services with requestors, would face problems of technical, security, and legal nature. From the technical point of view it has to be really easy for services to connect their websites with WoN, but it also has to be user friendly for common users. It has to work 24 hour a day and have quick support solving all high priority issues. Security is nowadays a problem for e-shops, but they hold the risk just for themselves. The matcher like WoN allowing users to buy anything through it and holding a lot of sensitive and private information about all services, carries a high risk. Also all the laws in all the different countries and the risk of fake accounts trying to steal money from honest users, means a lot of legal problems.

6.2 Architecture of WoN

Even though our own intelligent semantic matcher for Obziva.sk differs in many ways (see subchapter 6.4), before building it we had to analyze the architecture of WoN, to better understand problems we could face and what architecture we will use. After defining philosophy of WoN and also all the problems that could occur, we investigated creators final decisions about architecture of the matcher (see also Fig 6.1):

- De-centralization is the best way to go considering security, because there is not one entity holding all the information. For these purposes authors of WoN defined special structure called WoN nodes.
- Choosing RDF as data language for its flexibility.
- Needs are disconnected from their creator.
- Public descriptions for identification of person before initiating chat conversation.
- Anonymity is hard after creating world readable description, but at least no third party will be able to display all the needs of one user, since each need is separated.
- Messages are stored in WoN in RDF, when owners are offline.
- Other protocol layers to ensure security and integrity of data.

6.3 WoN and Obziva.sk

It is obvious that Obziva.s and WoN are based on the same principle of linking offers and demands. During semester in Vienna we had an opportunity to work with Florian Kleedorfer,

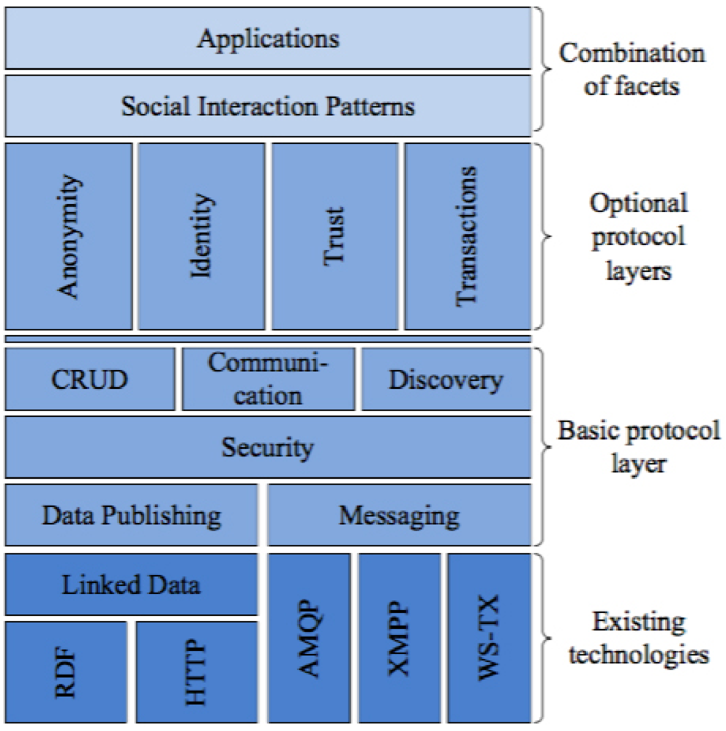


Figure 6.1: Diagram describing architecture of WoN taken from [35]

Christina Maria Busch and their team as part of mobility project. After the first meeting we agreed to connect Obziva.sk to WoN for mutual benefits. WoN benefited from testing its implementation process and accuracy of matching capabilities. On the other hand, we learned about semantics from professionals and in the future we can be one of the first portals that offering our services through WoN.

6.3.1 Implementation of Web of Needs

After studying and understanding advanced concepts of semantics and semantic matchers, we began implementation of WoN. To be able to post our offers to the matcher, we needed to run owner application which is programmed in Java programming language. This is where we came across many problems. Wiki [39] dedicated for documentation of WoN and instructions for its implementation was not complete and we were not able to make it running without authors help. Even with their advice and assistance we found many bugs which took developers several days to fix. After overcoming these problems, we renewed their documentation and made it easier to use for future services.

Next step was connecting to WoN server to be able to use our owner application. Since there is no actual online version yet, we had to use local server which was accessible only from research studios network. After connecting to server and the database of Obziva.sk we created queries that exported data from Obziva.sk, transformed it into RDF structure and posted them to local WoN server.

Unfortunately, because of its decentralization philosophy and limited owner functionality, we were not able to check whether all the data were posted correctly, but after using graphical user interface (GUI) as requestor we got matches from Obziva.sk without any problem.

6.3.2 Graphical user interface of WoN

To request imported services, we used WoN GUI, which did not work so well. At first we got stuck right after posting request, since the matcher returned nothing. We were recommended to use the newest GUI, but that did not work either along with user login. After a discussion with developers, they were able to resolve these problems in older user interface and began testing.

6.3.3 Testing matching capabilities of WoN

For testing purposes we used small amount of real demands(35), copied from Obziva.sk Facebook page. WoN assigns the value to the mach, indicating "how sure the matcher is that displayed match is correct for the request". Unfortunately this value is not available for requestor, and developers in Research Studio of Vienna were not able to display it without risking its dysfunction. Without this value we could not consider it as factor and examined each match, based on whether it is relevant to our request, or not. After testing was finished, it was clear that matcher is not working very good. For our 35 request we received 133 matcher of which 31 were the correct ones.

6.3.4 Why did we experience these complications?

Web of Needs is still in development phase and Obziva.sk was the first service to use this matcher. Many problems emerged, but that is what testing is for and it is safe to say that with authors personal help we managed to improve documentation and create a list of recommendation for future implementation:

- Matcher was comparing each part of need (title, description and tags) separately. Since users are used to write into tags the words that, according to them, should offer description contain, matcher should compare the tags with the description and the title.
- Installation was too complicated even for enthusiastic semantic developers, that could possibly want to implement WoN. The only solution is to use docker [40], a platform for distributed applications which contains all necessary technologies and configures them for potential service with one click.

- Create better and understandable documentation for developers based on assumption that they do not have deeper knowledge about how WoN works.
- Improve graphical user interface, allowing user to see value of the match or at least sort displayed matches by their value.

6.3.5 Future of WoN

Web of Needs is an interesting project and the whole experience in Research Studio of Vienna taught us much about the future of search engines. Google, Yahoo and other serach engines implement semantic search partially and it will be interesting to see their and WoN's progress over next few years. We believe that our contribution will help Florian Kleedorfer, Christana Bush and their team to improve matcher and seeing how driven they are, we have no doubt they can change the marketplace of the internet for better.

Chapter 7

Our matcher

Even though the work of WoN team was inspiring, relying on their matcher to solve our problem with demands was not an option. WoN matcher is not fully functional, could take them a long time to finish it and we would drive our visitors away from the Obziva.sk website. That is why we decided to create a semantic matcher of our own that would be intelligent and learn through time.

7.1 Collected data

As we already mentioned in chapter 5 the portal offers many services, categorized into categories and subcategories. Each user offering service has to be registered and have filled profile. This is to prevent the duplicity of offers which are now directly bonded with the user's profile.

After the incident that occurred after we started the portal, when everyone was offering everything, we decided to limit the number of offers to 5 per person and also make descriptions of offers required filed. This decision was made quickly, but after a year we can proudly say that no-one was complaining about this restrictions, even tough a lot of users are bypassing descriptions by filling random chars. It could seem like a problem but requiring descriptions is partially a way to discourage users from overloading Obziva.sk with fake offers and if someone fills description with random string, it just tells us that he is just lazy user, not a fake one. As of this day Obziva.sk offers more than 747 services of which only 206 have full descriptions. Since our matcher requires descriptions to work, after its deployment the users with fully filled profiles will be favored in search results.

Having these offers accessible we only needed demands for further testing purposes. As we learned during testing of Web of Needs matcher, extracting enough demands from our Facebook page would take too long, that is why we came up with creative questionnaire where participants had only 3 questions, each consisting of the select box containing list of all Obziva.sk subcategories and the text field. Instructions were straight forward: Select any desired category and create any request for service that should be in this category. Luckily we got little more than a 100 request queries of which we used 100 to test the matcher.

7.2 Stemming

Since we already have experience with stemmers from the semestral project, we decided that this time it would be best to create our own stemmer for Slovak language, instead of using Czech stemmer [41].

7.2.1 Slovak stemmer

To understand how Slovak stemmer works it is necessary to know most of the basic rules of Slovak language all of which are presented in official document Pravidlá slovenského pravopisu [42].

Verbs

Verbs are important part of service requests (demands) and all offers on Obziva.sk. After analyzing all the demands we saw that, when user is trying to find plumber he will intuitively write "I need someone to install my new tap." (in Slovak "Hľadám niekoho kto mi nainštaluje nový vodovodný kohútik."). Verbs in English would not be a problem since they have the same grammatical form for any personal pronoun except 3th person singular. On the other hand in Slovak we have different grammatical form for each personal pronoun the verb is referring to, also depending on whether it is singular, plural or generic. Verb conjugation is even more complicated considering the time the verb is referring to (present, past, future), which gives us a big set of suffixed to remove. To compare English and Slovak we present the table 7.1 with conjugated verb install for both in languages.

English - present	English - past	Slovak - present	Slovak - past
I install	I install**ed**	ja inštal**ujem**	ja som inštal**oval**
you install	you install**ed**	ty inštal**uješ**	ty si inštal**oval**
he install**s**	he install**ed**	on inštal**uje**	on inštal**oval**
we install	we install**ed**	my inštal**ujeme**	my sme inštal**ovali**
you install	you install**ed**	vy inštal**ujete**	vy ste inštal**ovali**
they install	they install**ed**	oni inštal**ujú**	one inštal**ovali**

Table 7.1: Comparison of verb conjugation install in English and inštalovať in Slovak

For better understanding of stemming process we highlight suffixes that stemmer should remove to get the root of the word.

Nouns

Nouns are even more complicated than verbs in Slovak language, which depends on gender, singular/plural case and morphological cases. Gender is divided into 4 groups: masculine - animate, masculine - inanimate, feminine and neuter. For each of these genders there are different models of declension. Each declension of these models changes by plurality and also by special morphological cases. Since it is really complicated, each case has two questions we ask to identify it:

- Nominatív - kto? čo? (who? what?)
- Genitív - koho? čoho? (who's? of what?)
- Datív - komu? čomu? (to who? to what?)
- Akuzatív - koho? čo? (who's? what?)
- Lokál - o kom? o čom? (about who? about what?)
- Inštrumentál - s kým? s čím? (with who? with what?)

After applying singular/plural and morphological cases we get for each model of decision 12 words that can differ from each other. For better understanding we present the list of all models along with different suffixes we have to remove by stemming, along with highlighted letters that are being replaces by these suffixes:

1. Masculine - animate
    - chlap : -a, -ovi, -om, -i, -ov, -om, -och, -mi
    - hrdin**a** : -u, - ovi, -om, -ovia, -ov, -v, -och, -ami
2. Masculine - inanimate
    - dub : -a, -u, -e, -om, -y, -ov, -om, -och, -mi
    - stroj : -a, -u, -i, -om, -e, -ov, -och
3. Feminine
    - žen**a** : -y, -e, -u, -ou, -ám, -ách, -ami
    - ulic**a** : -e, -i, -u, -ou, -iam, -iach, -ami

- dla**ň** : -ne, -ni, -ňou, -ní, -iam, -ne, -niach, -ňami
- kos**ť** : -ti, -ťou, -tí, -tiam
- gazdin**á** : -ej, -ú, -ou, -é, -ám, -ách, -ami
- ide**a** : -y, -i, -u, -ou, -í, -ám, -ách, -mi

4. Neuter

- mest**o** : -a, -u, -e, -om, -á, -ám, -ách, -mi
- srdc**e** : -a, -u, -i, -om, -ia, -iam, -iach, -mi
- vysvedčen**ie** : -ia, -iu, -ie, -í, -ím, -iam, -iach, -iami
- dievč**a** : -ťa, -ťu, -ati, -aťom, -atá, -at, -atám, -atách, -atami, -ence, -eniec, -encom, -encoch, -encami

We have to keep in mind that all listed are just regular ones and as we can see very little of these suffixes are unique for specific model. Since they repeat it would be too complicated to reconstruct each inflected word to its regular form. That is why for our purposes it will be sufficient to carefully remove these suffixes but without over-stemming the word.

Adjectives

After understanding nouns it is easy to learn how to inflect adjectives as we use the same morphological cases (Nominatív, Genitív, etc.). They are divided into 3 gender groups + plural and fortunately only 3 models:

1. Masculine

- pekn**ý** : -ého, -ému, -om, -ým
- cudz**í** : -ieho, -iemu, -om, -ím
- otcov : -ho, -mu, -om, -ým

2. Feminine

- pekn**á** : -ej, -ú, -ou
- cudz**ia** : -ej, -iu, -ou
- otcov**a** : -ej, -u, -vou

3. Neuter

- pekn**é** : -ého, -ému, -om, -ým

- cudz**ie** : -ieho, -iemu, -om, -ým
- otcov**o** : -ho, -mu, -o, -om, -ým

4. Plural

- pekn**é** : -í, -ých, -ým, -é, -ými
- cudz**íe** : -í, -ích, -ím, -ie, -ími
- otcov**e** : -i, -ých, -ým, -ých, -ými

There are also comparative forms which have only two suffixes -ší and -ejší and superlatives created by prefix naj-. As we found out from our questionnaire these forms are never used when searching for service and stemmer can omit them without any compromise.

Over-stemming

Some stemmers do not consider cases where the words get deformed by the process of stemming, since these exceptions are hard to detect. Over-stemming occurs when we remove the part of the word which is not suffix and these exceptions were cleverly solved by Hana Pifková [43] in her own stemmer. She noticed that if the word ends by l',l,r or ŕ it is probably case of over-stemming and we have to add the removed part of the word back. Since she was focusing only on nouns we tried to upgraded her solution and added other exceptions for endings like š, č, ž, é, á and other characters with diacritics. To make sure that non of important words end with these characters we used Words-finder [44]. Even though it did not bring any advantage, we did not notice any disadvantage and kept it that way.

Process of stemming

After studying Slovak grammar thoroughly and defining all the suffixes, the process of stemming was simple and straight forward. First stemmer tries to remove the longest suffix and if the word does not end with it, stemmer continues by trying other suffixes. The reason for trying the longer ones first is that some of the suffixes contain other suffixes and we want to remove the longest part of the word. If over-stemming occurred we undo this change and continue through our list of suffixes. Someone could suggest that we iterate stemming script like the Husk's stemmer does, until the word stays the same for the whole iteration, but on our testing data-set we did not notice any improvement. That is why we decided that it is not necessary and it could only slow the whole stemming process.

7.2.2 Offers

The stemmer was finished and we started to think how to store stemmed versions of offers in the database. When we compared all the other search options in chapter 3, we realized that full-text search has advantage of indexing and it can be easily implemented along with stemming. Based on these assumptions we created separate table with all the stemmed words as indexes to offers containing them. Fortunately we quickly realized that for later evaluation of match we will use offers description to get to the stemmed words and going through all the words and checking whether they point to selected offer or not would be inefficient. For this reason we created one more column with all stemmed words of the offer stored in JSON [10] format. In addition we also added stemmed names of category and subcategory corresponding with offer as these words are not necessarily used in descriptions, even though the requestors are searching for them.

7.2.3 Demands

To guarantee the words' consistency in offers and demands, and to make searching for match easy, each requestor's query is stemmed. Having all the stemmed words of the demand, our matcher can search for them in words database table and return all the matching offers. For further purposes we also save all demands to the database with their stemmed form in separate column.

7.3 Matching

Returning all the offers that have only one word in common with demand would of course result into a lot of bad matches, unless we count how many words they have in common and display the best at the top of the search.

7.3.1 Unimportant words

Unfortunately the fact that two texts have few words in common does not mean that they have meaning correlation, especially when they are just conjunctions. To avoid this problem we created a bag of words [3] to store all the expression that are not important for our search and modified the stemmer to skip them before inserting into database.

7.3.2 Important words

Removing unimportant words from matching process did not guarantee that matches will be relevant, since there are other words like "I", "searching", etc. that have no context value. To make sure each word has a value, reflecting its importance, we implemented word evaluation method TF-IDF from subsection 4.2.

As the name suggest we need to compute two values: term frequency and inverse document frequency. In out case term frequency (TF) means number of times that word occurs not only in all the offers but also in all the demands, which is why it was important so store them. Inverse document frequency (IDF) is computed by following formula:

$$IDF = log(\frac{D}{dt})$$

where D represents the count of all offers and dt is the number of offers containing the word. The final value is calculated by multiplying term frequency with inverse document frequency. As TF-IDF can vary from 0 to any value, which was in our case from 6.5 to 120, we used Feature scaling [45] method to normalized all the values to range from 0 to 10:

$$f(x) = \frac{(newMax - newMin) * (TFIDF - oldMin)}{oldMax - oldMin} + newMin$$

After the normalization of TF-dif we were able to define **the matching certainty**, the value reflecting how sure the matcher is that the match is a good one for the search query, by adding all the TF-IDF values of words that offer and demand have in common.

7.4 Intelligent matcher

Users are the best judges when it comes to search results. It is not rare that the user does not click on every search suggestion for his query and sometimes not even on the first ones. He decides judging by the descriptions whether suggestion fulfills his searching need or not. This type of data is very important for search engines since they can learn from it. We know from our own experience that searching for the same query after few months does not mean that we will get the same suggestions.Their position in search results depends a lot on visits of users and each time the user clicks on a suggestion, he makes its position stronger compared to others.

7.4.1 Relationships between words - Graph solution

We wanted to apply the similar principle as search engines use for visitations to our matcher, but give users even more control over semantic matching based search by allowing them to mark the match as correct or incorrect. This could lead to several problems including abuse, which is why we had to create an algorithm, that slowly changes the stored matching certainty and grows faster by the number of evaluations, creating a relationship between query and match. As we implied this relationship can be positive or negative but it should not bind demand and offer as such, since there is very little possibility that someone will write the same query as the users before. That is why we decided to create graph like structure where each node represents a word and links between them represent their relationship. This way we can create relationships between the words from the demand and the words from the offer, making it applicable to any future queries.

7.4.2 Learning

The user's input is obviously very important for our matcher to be able to learn. In machine learning it is called learning with supervisor where in our case the user is the teacher. As we mentioned earlier it is important to create these relationships carefully and adjust teaching speed by the number of evaluations.

First we defined scale by which we will adjust teaching speed. We have to keep in mind that there are positive and negative evaluations and the absolute value of the difference between them is what is important for computation of relationship. For practical reasons our relation scale goes from 0 to 100 points as it represents percents of how sure the matcher is that the relationship is negative or positive. Any higher relation difference is not necessary for computational purposes.

After defining the scale we created learning formula. It was important that has maximum and minimum value which is logarithm. We chose logarithm with the base 2 and normalized evaluation difference to rage 1 - 5 using Feature scaling as this value is the parameter for logarithm, providing us final results ranging from 0 to approximately 2.322. Again we normalized the result to range 0 - 1, which would not be possible without knowing the heights and the lowest possible value of result. The reason for normalization of the final result to this range is explained later.

There was still a problem of logarithm growing too fast at the low number of evaluations and not fast enough when the number was high. We resolved this issue by modifying final normalization by increasing its highest value from 0 to 1 by 0.1s, depending on number of evaluations. Here we present the final steps of our computational formula for learning where **pt** is number of evaluations:

1. normalize **pt** to range from 1 to 5
2. compute **logarithm** = log2(**pt**)
3. compute the **ceil** of normalization = (round up **pt** to 10s)/100
4. normalize **logarithm** from range 0 - 2.322 to 0 - **ceil**

The final formula being:

$$f(pt) = normalize(log2(normalize(pt, 1, 100, 1, 5)), 0, 2.322, 0, round(normalize(pt, 1, 100, 1, 5), 10))$$

Having a learning quotient ranging from 0 to 1 we could finally compute the relationship between words by multiplying the arithmetic mean of both TF-IDF word values with the learning quotient. The reason for arithmetic mean is that it is not possible to say which of the two words should be used for this computation, also keeping in mind that by choosing offer's word or demand's word we could cause unpredictable change of the relation value. This could happen when in the first search the word number 1 would be in the demand, number 2 in the offer and in another search they would be reversed.

At this point, the range of learning quotient makes sense, since it assigns the full value of the mean to word relationship only if it the value of evaluation difference is 100 positive. When the difference is negative the final value is multiplied by -1, resulting into negative relationship.

Now when the matcher compares offer and demand all the relations are added together along with the same words which always have the full TF-IDF value.

7.5 Final modifications

Slovak language is based on Latin script using diacritics like in following charactes: č, é, ô and ä. Even though diacritics should be used according to Pravidlá slovenského pravopisu [42], after collecting all the testing data for requests we noticed that some of them are written without diacritics. This could result into bad stemming of the words, but it is not possible to reconstruct them into form with diacritics. On the other hand we were able at least adjust matching, by removing the diacritics of the words from database before comparing them with a word from demand.

Chapter 8

Testing and results

The matcher was ready and together with testing data from the online questionnaire, providing us 100 queries with assigned categories, we could start designing the testing conditions.

8.1 Testing setup

Since we already tested WoN's capabilities, we knew that the first way to evaluate the matcher was:

1. request matches for all the demands
2. for each match decide whether it is correct or incorrect
3. count all correct and incorrect matches
4. compare results to get matcher's success rate.

Unfortunately this was the only possible type of testing for the Web of Needs, since the matcher did not display matching certainty, neither it did not sort the matches by this value. Our matcher contains this feature which allowed us to evaluate its capabilities by the second test: counting only the top three matches.

After the testing of simple matching which used stemming and tf-idf evaluation, we began learning of matcher by acquired demands from questionnaire. Each time the match was evaluated we used learning function to create positive or negative relationships between the words from the offer and the demand. Since the matcher learns more quickly as the number of evaluations gets higher, we ran this evaluation process 50 times, to make sure that relationships between words are strong and relevant. When the learning process was finished we ran the same evaluation of correct and incorrect matches from the first testing.

8.2 Threshold

After the first test runs we realized, that we cannot display all the matches, especially when it has low matching certainty. To solve this problem we added restriction to display only those results, that are higher than specified threshold - which is in our case 12. This number is just based on our assumption after few test runs and does not have any scientific explanation.

8.3 Results

The first testing method returned 170 matches of which 65 were correct and 105 incorrect, resulting into 38% success rate. Even though we had smaller testing data set for Web of Need, it was possible to calculate its success rate and compare it with Obziva's stemmer. As we can see in Fig. 8.1 even basic version of our matcher without learning returns 15

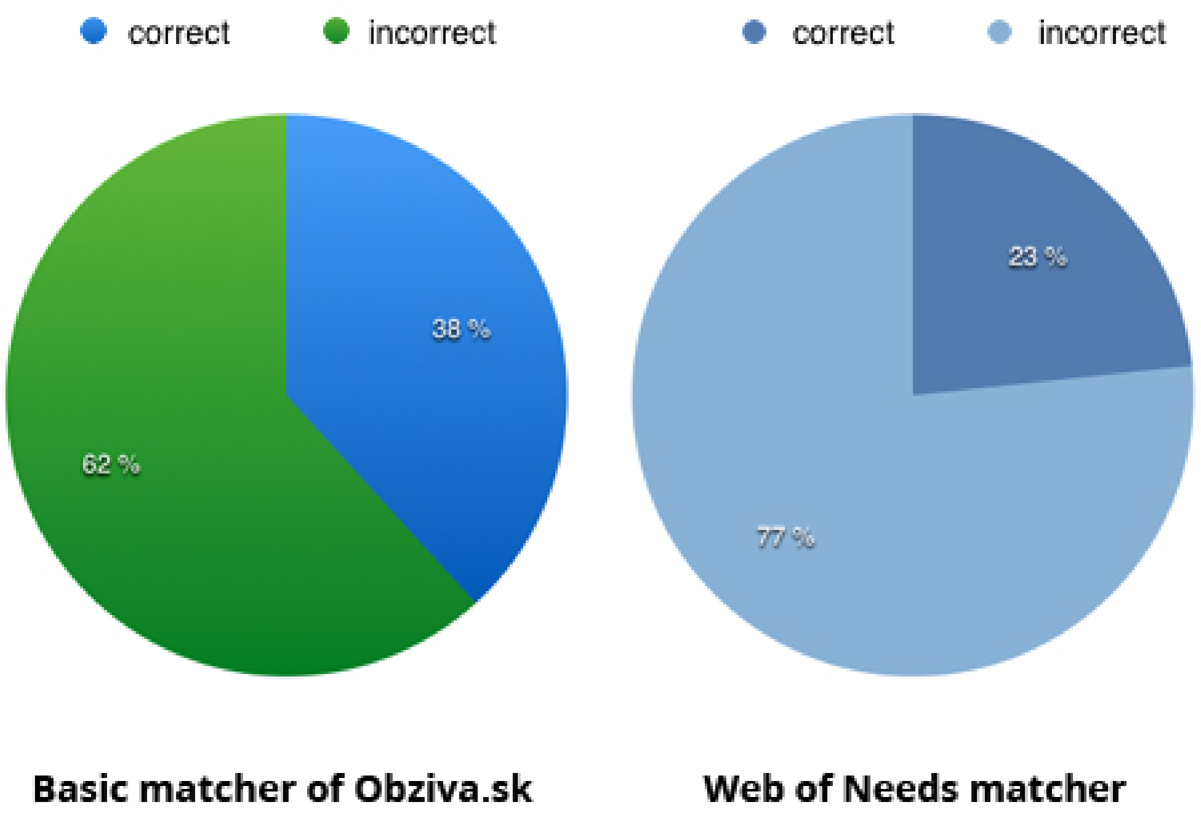


Figure 8.1: Comparison of correct and incorrect matches of Obziva's stemmer and WoN stemmer

The second testing round was focusing on evaluation of matcher, considering only the top three matches. The number of matches was reduced to 124, but the results were little bit worse than with the first testing method, evaluating the matcher's correctness by 35%.

Before the last test we had to run learning function based on testing data. The learning algorithm created more than 7,000 relationships between the words and iterated 50 times. After learning stemmer all the relationships between words, we examined once again its matching accuracy by repeating the same process as in second round of testing. The results showed that the number of matches was reduced to 38 of which 18 were correct, resulting into the success rate of 47%. The explanation why the number of results was highly reduces can be found in following chapter.

For better comparison of the matching accuracy before and after learning see Fig. 8.2

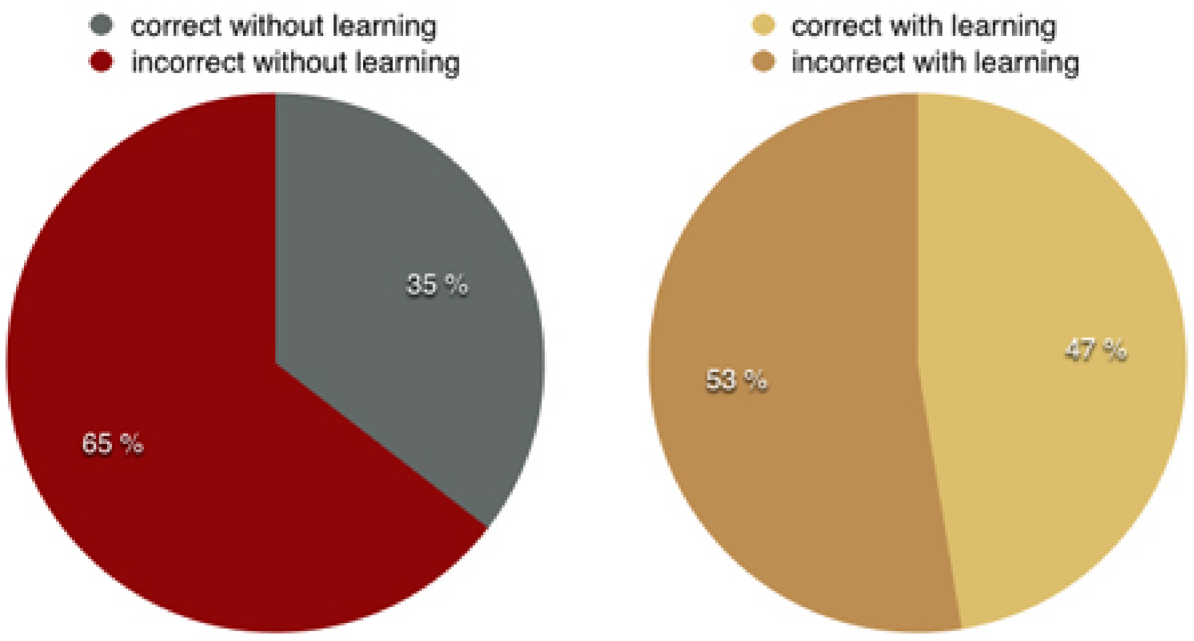


Figure 8.2: The matching success rate of matcher before and after applying supervised learning method

Chapter 9

Discussion

During the whole process of creating our intelligent semantic matcher we found many simple problems that were solved right away and also more complicated ones we discovered after testing. In this chapter we propose some solutions to these problems for the future implementation.

9.1 Stemming

Creating our own stemmer was a challenge as we had to study a lot of rules of Slovak language. Even though we are very pleased by its functionality and results it provides there is always a room for improvement.

The first suggested improvement for the future is to stem also prefixes. Prefixes are tricky and some can change the whole meaning of the word, but others can be removed without the loss of meaning.

The second modification could be reconstruction of words. Our stemmer does not reconstruct the word after removing its suffix, which means we are partially loosing its semantic value. The majority of the stemmed words, in the form as they are now, are hard to read for humans and even would be useless for other services.

The last improvement of the stemming could be realized by solving the problem of diminutives. At first we thought that people will not use them in search queries, but as it turns out there were 3 demands containing a word in diminutive form.

9.2 Search queries

The problem with search queries is that users use slang or other expression, which are not a part of the official rules and recommendations of the language. On the other hand the matcher such as ours can match even these requests, if there is any offer containing the same slang words.

The much more complicated issue is when users do not use diacritics. This can easily cause over-stemming or under-stemming and the matches will not be as good as if the query was written with diacritics. The solution of this problem could be a reconstruction algorithm finding patterns and repairing users mistake.

9.3 Normalization

As we could see in section 7.3. there is a problem with normalization where the term frequency - inverse document frequency does not have a ceil and can change with the new offers of demands. That is why for its normalization we had to manually input the highest value into the normalizing function, which makes it unsustainable or at least inaccurate. For this purposes it is possible to implement the algorithm, recalculating the TF-IDF values regularly and normalizing by the highest value we get from the database.

9.4 About results

We were amazed to find out that not only our matcher is better than WoN matcher, but in its final form it is twice as accurate. Unfortunately the results indicated a high reduction of matching results after applying the learning function of the matcher and filling it with testing data. The closer examination showed that the reduction was caused by the less important words. Even though their TF-IDF was lower, the numerous occurrences resulted into their multiple evaluations, making them crucial for matching process. The only solution we came up with is to create new learning rules that omit the words importance, based on its value and the number of evaluations, meaning that if the TF-IDF is low and evaluated too many times, it should not be considered by matcher.

9.5 Final word

In this discussion we considered all the advantages, disadvantages and future updates of our stemmer. We have a strong believe that our intelligent semantic matcher works very good and thanks to the newest methodologies such as machine learning, stemming and information retrieval method TF-IDF it has a great potential in the future.

Bibliography

[1] Sebastian Löbner, *Understanding Semantics*, London : Arnold, 2013

[2] Dean Allemang, James Hendler, *Semantic Web for the Working Ontologist*, 2008
`http://workingontologist.org/`

[3] Charu C. Aggarwal, ChengXiang Zhai, *Mining Text Data*, 2012

[4] C.Ramasubramanian, R.Ramya, *Effective Pre-Processing Activities in Text Mining using Improved Porter's Stemming Algorithm*, International Journal of Advanced Research in Computer and Communication Engineering Vol.2, Issue 12, December 2013

[5] Anvitha Hegde, Mrs Savitha K Shetty, *A Study on Stemming Algorithms*, International Journal of Emerging Trends in Science and Technology, 2015

[6] Martin Porter, *Wikipedia - Stemming*
`https://en.wikipedia.org/wiki/Stemming`

[7] Martin Porter, *Snowball - language for writing stemming algorithms*
`http://snowball.tartarus.org`

[8] Julie Beth Lovins, *The Lovins stemming algorithm*, 1968

[9] P. Majumder, M. Mitra, S. Parui, G. Kole, *YASS: Yet Another Suuffix Stripper*, 2006

[10] Wikipedia page about JSON format
`https://en.wikipedia.org/wiki/JSON`

[11] *Microformats*, 2005
`http://microformats.org/wiki/about`

[12] Mark Birbeck, Steven Pemberton *RDF/A Syntax*, October 2004
`https://www.w3.org/MarkUp/2004/rdf-a.html`

[13] Ben Adida, Mark Birbeck,Shane McCarron, Steven Pemberton *RDFa in XHTML: Syntax and Processing*, 2008
`https://www.w3.org/MarkUp/2008/ED-rdfa-syntax-20081004/rdfa-syntax.pdf`

[14] *WHATWG*
`https://whatwg.org/`

[15] Alexis Goldstein, Louis Lazaris, Estelle Weyl, *HTML5 & CSS3 FORTHEREAL WORLD*, 2011
`http://goo.gl/ET9pYB`

[16] WHATWG *HTML Living Standard*, 2016
`http://www.whatwg.org/html/`

[17] *World Wide Web Consortium (W3C)*, 2016
`https://www.w3.org/`

[18] W3C *HTML 5.1*, 2016
`http://dev.w3.org/html5/spec/`

[19] *Public dicussion about HTML5 and RDFa*
`http://goo.gl/fvLqRN`

[20] Dr. Jenifer Fay Alys Tennison, *HTML5/RDFa Arguments*, August 2009
`http://www.jenitennison.com/2009/08/21/html5-rdfa-arguments.html`

[21] *Ian Hickson - wikipedia*
`https://goo.gl/OkTVCN`

[22] *Schema.org*
`http://schema.org/`

[23] *Microdata - itemtype Movie*
`http://schema.org/Movie`

[24] Ivan Herman, *SPARQL is a Recommendation*, January 2008
`https://goo.gl/okarwm`

[25] Jin-Xing Hao, Angela Yan Yu, Ron Chi-Wai Kwok, *A Semantic Analysis Method for Concept Map-based Knowledge Modeling*, School of Economics and Management, Beihang University, Beijing, China and Department of Information Systems, City University of Hong Kong, Hong Kong, China, December 2010

[26] *WordNet*
`https://wordnet.princeton.edu/`

[27] *SAShE.sk - slovenský handmade design.*,
`www.sashe.sk`

[28] Knuth, Donald Ervin, *The art of computer programming*, Reading Mass. : Addison-Wesley, 1997

[29] Diane Nahl, Violet H Harada, *Composing Boolean Search Statements: Self-Confidence, Concept Analysis, Search Logic, and Errors*, School Library Media Quarterly, v24 n4 p199-207, 1996

[30] Hannah Bast, Florian Bäurle, Björn Buchhold, Elmar Haussmann *Broccoli: Semantic Full-Text Search at your Fingertips*,Department of Computer Science University of Freiburg 79110 Freiburg, Germany, July 2012

[31] Stuart J. Russell and Peter Norvig, *Artificial intelligence: a modern approach*,Englewood Cliffs, N.J. : Prentice Hall, 1995

[32] L'udovít Malinovský, *Kategorizácia vysokorozmerných dát pomocou rozlišovacích kritérií.*, conference paper, May 2010

[33] Juan Ramos, *Using TF-IDF to Determine Word Relevance in Document Queries*, Department of Computer Science, Rutgers University, 23515 BPO Way, Piscataway, NJ, 2003

[34] Eetu Mäkelä, *Survey of Semantic Search Research*,Proceedings of the Seminar on Knowledge Management on the Semantic Web, Department of Computer Science, University of Helsinki, 2005

[35] Florian Kleedorfer, Christina Maria Busch,Christian Pichler, Christian Huemer, *The Case for the Web of Needs*,16th IEEE Conference on Business Informatics (CBI 2014), Geneva, Switzerland, July 2014

[36] Florian Kleedorfer, *Building a Web of Needs*. 10th International Semantic Web Conference, Outrageous Ideas Challenge, Bonn, Germany, November 2011

[37] Florian Kleedorfer, Christina Maria Busch , *Beyond Data: Building a Web of Needs*. In Proceedings of the WWW2013 Workshop on Linked Data on the Web (LDOW 2013), Rio de Janeiro, Brazil, May 2013.

[38] Florian Kleedorfer, Christina Maria Busch, Gabriel Grill, Soheil Khosravipour, Fabian Salcher, Alan Tus, Erich Gstrein, *Web of Needs-A New Paradigm for E-Commerce*. Business Informatics (CBI), 2013 IEEE 15th Conference on , vol., no., pp.316,322, 15-18 July 2013

[39] *Wiki documentation for WoN*
`https://goo.gl/vRg4Zi`

[40] *Docker website*
`https://www.docker.com/`

[41] Ljiljana Dolamic, Jacques Savoy, *Indexing and stemming approaches for the Czech language* in Journal Information Processing and Management: an International Journal archive, volume 45, issue 6, 2009, pp. 714-720

[42] prof. PhDr. Ján Horecký; DrSc. PhDr. Viktor Krupa, DrSc.; Ing. Vladimír Benko; Mgr. Vladimír Radik; Mgr. Marta Zamborová; PhDr. Jitka Madarásová and others; *Pravidlá slovenského pravopisu*, Slovenská akadémia vied - Jazykovedný ústav Ludovíta Štúra, 2000

[43] Hana Pifková, *Slovenský stemmer*
`http://goo.gl/km4rNz`

[44] Words-Finder
`http://sk.words-finder.com`

[45] Feature scaling - Wikipedia
`https://goo.gl/D51ZXI`