TALLINN UNIVERSITY OF TECHNOLOGY
Faculty of Information Technology
Department of Informatics

IDU40LT

Kristin Ehala 134637 IABB

# FINDING SUITABLE TERMS FOR QUERY EXPANSION USING WORDNET DICTIONARY BASED SEMANTIC SIMILARITIES

Bachelor's thesis

Supervisor: Gunnar Piho
PhD
Associate Professor

Ahti Lohk
PhD
Teaching Assistant

Tallinn 2016

TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond
Informaatikainstituut

IDU40LT

Kristin Ehala 134637 IABB

# PÄRINGU LAIENDAMISEKS SOBIVATE TERMINITE OTSIMINE, KASUTADES WORDNET SÕNASTIKUL PÕHINEVAID SEMANTILISI SARNASUSI

Bakalaureusetöö

Juhendajad: Gunnar Piho
Doktorikraad
Dotsent

Ahti Lohk
Doktorikraad
Assistent

Tallinn 2016

# Author’s declaration of originality

I hereby certify that I am the sole author of this thesis. All the used materials, references to the literature and the work of others have been referred to. This thesis has not been presented for examination anywhere else.

Author: Kristin Ehala

22.05.2016

# Abstract

Internet users frequently use incomplete query terms or human language near sentence construction to search for data from search engines. In this thesis three methods are proposed for finding suitable terms for query expansion (QE) using WordNet dictionary based measures of similarity. QE is a process where the user input query term or words are reformulated to improve the query results, for example in search engines, and to match additional documents and pages. The terms the system suggests are semantically related to the initial query word and found using synsets within lexical database WordNet. The three measures used are unified semantic similarity measure, which is combination of the shortest Path based measure, Wu & Palmer's based measure and Leacock & Chodorow's Measure; definition based similarity, which uses definitions of synsets in WordNet and a hybrid method, where both previous methods are combined. All the methods are analysed and compared to each other. It is stated in this thesis, that the most useful method for finding recommendable terms for query expansion is the combined hybrid method. Some key points that need to be taken into account when working with dictionary based similarity measures are also presented.

This thesis is written in English and is 41 pages long, including 5 chapters, 2 figures and 5 tables.

# Annotatsioon

# Päringu laiendamiseks sobivate terminite otsimine, kasutades WordNet sõnastikul põhinevaid semantilisi sarnasusi

Internetikasutajad satuvad tihti olukordadesse, kus erinevaid otsingumootoreid kasutades ei leita vasteid või on vasted ebapiisavad. See on tingitud sellest, et inimesed kasutavad otsinguks loomulikku keelt, väga spetsiifilisi sõnu või ei arvesta nad sünonüümide ja lauseehitusega. Antud lõputöös esitab autor kolm meetodit semantiliselt sarnaste sõnade leidmiseks, et laiendada päringuid. Päringu laiendamine on protsess, mille käigus sõnastatakse kasutajate sisestatud päringud ümber, seeläbi soodustades sobiva info ja dokumentide leidmist. Lõputöös kasutatakse WordNet sõnastikul põhinevaid semantilisi sarnasusi. WordNet on inglise keele leksikaalne andmebaas, mille on käsitsi koostanud Princetoni kognitiivse teaduse labor. Kolm semantilise sarnasuse leidmise meetodit, mida töös rakendatakse, on semantiline sarnasus mõistete kauguste kaudu, semantiline sarnasus mõistete definitsioonide abil ning hübriidmeetod, kus kaks eelnevat meetodit kombineeritakse. Mõistete kauguse meetodis kasutatakse kolme juba olemasolevat teedel põhinevat meetodit, milleks on kõige lühema tee meetod, Wu & Palmer-i meetod ning Leacock & Chodorow meetod. Definitsioonil põhinev meetod võtab sõnade sarnasuse võrdlemisel kasutusele WordNeti pakutavad definitsioonid. Sõnad on üksteisele tähenduslikult lähedased, kui nende definitsioonid sisaldavad kattuvaid sõnu. Kõiki meetodeid analüüsitakse eraldi ning ka omavahel. Töö käigus selgitatakse välja parim meetod semantiliselt sarnaste sõnade leidmiseks, kuid tuuakse välja ka võrreldavate meetodite positiivsed ja negatiivsed küljed. Töös esitatavas süsteemis leitakse mõistetele sarnased sõnad ükshaaval. Kõige kasulikumaks meetodiks osutub hübriidmeetod, kus rakendatakse koos kaugustel ning definitsioonidel põhinevat sarnasust. Hübriidmeetod väljastab soovitavate sõnade minimaalse hulga, kus suurem osa sõnadest on originaalsele otsingusõnale semantiliselt lähedased. Töös tuuakse välja mitmeid erinevaid probleeme ja mõttekohti, mida tuleb arvestada loomuliku keele töötlemisel. Samuti käsitletakse konkreetselt WordNeti puudutavaid asjaolusid. Lisaks teeb autor töös ettepanekuid, kuidas meetodeid parandada ning mida süsteemi arendamisel tähele panna.

Lõputöö on kirjutatud inglise keeles ning sisaldab teksti 41 leheküljel, 5 peatükki, 2 joonist, 5 tabelit.

# List of abbreviations and terms

| | |
|---|---|
| CET | Candidate expansion term |
| Homonym | One of a group of words that share the same pronunciation but have different meanings |
| Hypernym | More general term of a group of terms |
| Information filtering systems | A system that removes redundant or unwanted information from an information stream using (semi)automated or computerized methods prior to presentation to a human user |
| LCH | Leacock & Chodorow's Measure |
| NLP | Natural Language Processing |
| QE | Query expansion |
| CET | Candidate expansion term |
| NLP | Natural Language Processing |
| NLTK | Natural Language Toolkit |
| PATH | The Shortest Path based Measure |
| QE | Query expansion |
| RS | Recommender system |
| Serendipity | Luck that takes the form of finding valuable or pleasant things that are not looked for |
| Stop words | In computing, stop words are words which are filtered out before or after processing of natural language data (text). Stop words usually refer to the most common words in a language |
| SWSNL | Semantic Web Search Using Natural Language |
| Synset | A set of one or more synonyms that share a common meaning |
| Taxonomy | The practice and science of classification of things or concepts |
| WSD | Word-sense disambiguation |
| WUP | Wu & Palmer's Measure |

# Table of contents

# List of figures

# List of tables

# 1 Introduction and Background

The importance of information technology and Internet has increased rapidly during the past 30 years. We are overloading with the information that is approachable from the Internet. To some extent, the problem of having too much information, is partly being resolved with the usage of search engines, which provide us with useful information that we are searching for. They give the user an exact information he or she wanted, but leave little room for new discoveries.

This problem of excess information is also slightly relieved with recommender systems (RS). Recommender systems are software tools and techniques, whose goal is to make useful and sensible recommendations to a collection of users for items that might interest them. This means that RS helps users to find similar items but not the same exact ones. The most popular fields that use RS include music and film industries and also e-commerce, e.g. Spotify, Netflix, YouTube and Amazon. Recently there has been an increase in the usage of RS in social media pages and applications, e.g. Twitter, Facebook, LinkedIn, Tumblr and Tinder etc. recommending interests, pages, followers and even people on online dating sites. But when these kinds of recommender systems evolve, the problem of over-specialization arises [1]. This means that RSs are more focused on recommending same things all over again and do not take into account products, pages etc. that are new and don't have enough rated data about themselves yet. Due to that new content is not being recommended for users.

When talking about random findings on the Internet, serendipity plays a meaningful role. Serendipity is the act of making fortunate discoveries by accident. It has had a significant meaning in the past of science, technology and many other fields. [2] As mentioned before, nowadays most of our Internet behaviour is somehow recommended to us and there is little room for completely new discoveries. There are still some pages that allow us to do so. StumbleUpon is one of the websites that tries to pursue the "unexpected encounters" of Internet browsing. Similar web pages are mostly focused on randomly

selecting from the large set of websites and not so much focusing on the content based similarities.

One other way to enhance “unexpected encounters” in our everyday Internet behaviour is to use query expansion (QE) in search engines. Query expansion is a process where the user input query term or words are reformulated to improve the query results. When talking about QE, users may frequently provide incomplete query terms or use synonyms that are not used for actually finding the results. [3] The process of QE is a widely used technique. It takes into account the user input query, uses some methods, usually text analysis or natural language processing, and finds candidate expansion terms (CETs). The search for suitable query results uses both the user input query and all the CETs. This broadens search results with texts and documents that could also interest the user.

## 1.1 Task

The aim of this thesis is to propose a suitable way to find recommendable candidate query expansion terms (CETs) using WordNet dictionary based semantic similarity methods. In this thesis it is assumed that the query input, the user chooses, is one word, not a full question or a sentance. The recommended CETs have to be similar or have a related meaning to the user query, so that humans would also see the relatedness between the words.

This thesis has the following objectives:

1. To bring out examples of path based semantic similarity measures provided in collaboration with WordNet and Python Natural Language Toolkit and to find a way to use them most efficiently for recommending CETs.

2. To use word definition based similarity to find most suitable CETs for QE.

3. To analyse which methods of this thesis work best for finding suitable candidate expansion terms.

4. To bring out the problem areas and ideas to consider when finding suitable CETs and when working with WordNet and natural language processing.

## 1.2 Approach

To reach the target objective, I am going to use already existing database WordNet. WordNet is a large lexical database on English language, where nouns, verbs, adjectives and adverbs are grouped into sets of cognitive synonyms (synsets), each expressing a distinct concept. [4] To analyse the semantic relatedness, I am using programming language Python version 2.7.11. I chose Python because of previous experience and because it has a Natural Language Toolkit (NLTK) package, which allows the users to create programs that perform tasks with human language data using WordNet database [5].

NLTK has variety of different built-in semantic similarity measures, which compare semantic-relatedness between two synsets [6]. In this thesis I am using three WordNet edge-counting based measures: Shortest Path, Wu and Palmer and Leacock-Chodorow's similarity. Edge-counting similarities are being chosen to compare two different methods when finding CETs. The measures are explained in the theoretical background chapter.

To compare the accurateness of recommended words, I am also finding possible CETs with the help of the definitions of the query words. The hypothesis is that if compared words have 2 or more common words in their definitions, these compared words are most likely related.

To refine the results and to reduce the possible number of matches, I am also creating a combined method of both semantic similarity measures and definition based similarities, and compare this to the individual methods.

Four features that distinguish my work:

1. Three edge-based similarity measures are transformed into one scaled measure that takes into account the observation results of each individual semantic similarity measure and scales the results accordingly.

2. The similarity of two compared words is measured using the definitions of the words.

3. Three edge-based similarity measures and the definition based similarity will be made into a combined hybrid method.

4. The similarity measures and definition based similarity is used only on the nouns of the WordNet database.

## 1.3 Overview of Thesis

In the Related Works section I present researches about the topic and how other people have discussed and analysed this issue and methods.

In the theoretical background paragraph I am going to explain WordNet and how the relationships between words and their meanings are built. I also explain Python NLTK and its semantic similarity measures for WordNet.

In the experiments section I describe the process of finding suitable CETs using only semantic similarity measures, definition based measure and then the measures combined.

I also bring out the main problems that occur with this type sematic similarity recommendations and conclude with the results and analysis and recommendations for future work.

The work is carried out in the Tallinn University of Technology for a Business Information Technology bachelor's thesis in the spring of 2016.

# 2 Related Works

There are many researches based on WordNet similarity measures. Many of them use WordNet to improve query expansion, others to analyse semantic similarity. Most popular methods are edge-counting based and corpus based [7]. Meng, Huang and Gu brought out nine different approaches in their research about WordNet Semantic Similarity Measures, but there are also a high variety of hybrid methods used. [8]

The idea of automated QE reaches back to the 1960 [9]. Most of the previous WordNet related QE researches use WordNet itself to take the CETs from. One difference that I encountered was the research of Pal, Mitra and Datta. They used WordNet similarity measures to improve QE, but CETs were gathered from various input texts, not from WordNet directly [9]. Zhang, Peng and Li (2009) added synonyms to the set of possible query words to expand the result set. Fang (2008) selected CETs on the principle where the vocabulary overlaps between CETs definition and the definition of a query term [9].

Harabagiu and Moldovan brought out an example in their research "Knowledge Processing on an Extended WordNet". In their example, a path between "hungry" and "refrigerator" is established because the markers collide at the node "food", which is part of the semantic network of both "hungry" and "refrigerator". Thus, an interface between being hungry and going to the refrigerator can be made [10].

The idea for definition based similarity, where definitions have common terms, originates from Lesk Algorithm. This is related to word sense disambiguation (WSD) problem. The original Lesk algorithm was used to disambiguate a target word by selecting the sense whose definition has the largest number of words that overlap or match with the definition of neighbouring words [11]. This method is used in texts and sections, and not for single words.

# 3 Theoretical Background

In this section I briefly give an overview of WordNet, semantic similarity and relatedness, natural language processing with Python Natural Language Toolkit and dictionary based similarity measures.

## 3.1 About WordNet

WordNet is a manually constructed English language database created by Cognitive Science Laboratory of Princeton University. It arranges nouns and verbs into set of synonyms (or synsets or lexicalized concepts), gives a short definition of the synset and is able to provide related antonyms and synonyms for its members (words). Synsets consist of words with cognitively the same meaning. They can be interlinked by means of conceptual-semantic relations, such as hypernymy, hyponymy, holonymy, meronymy and, other relations [4]. All noun and verb synsets of the WordNet are organized into the separate hierarchies of hypernymy-hyponymy (or super-subordinate or IS-A) relations that can be presented as trees. The root concept of the noun hierarchy is “entity” and all the other nouns are a part of this hypernym. The graph below shows the start of the IS-A hierarchy of noun in WordNet“. (Figure 1).

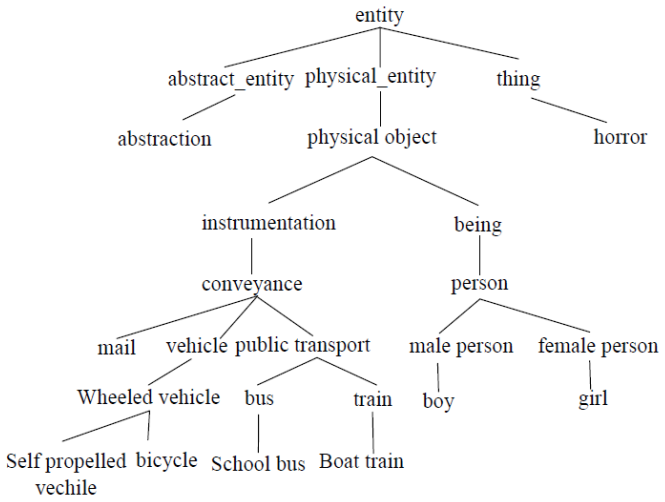


Figure 1 A Fragment of a noun IS-A relation in WordNet [8]

WordNet also has similar hierarchy for verbs, but verbs do not have only one root concept for the hierarchy. In 2013 a verb hierarchy with only three unique beginners: *to be, to do, happen,* was proposed [11].

WordNet also contains other lexical classes, such as adverbs and adjectives, but those items are not organized in a IS-A hierarchy. [12]

WordNet displays each noun synset in a form of "word" + 'n' + "01-*"

- "word" is any noun in English language.
- "n" means that the "word" belongs to a noun lexical class.
- "01" or "02" etc is used to define the different meanings of the same synset. An example is shown below. Each "engineering.n.*" represents different meaning of the same lexical word form "engineering".
    - Engineering.n.01 - the practical application of technical and scientific knowledge to commerce or industry
    - Engineering.n.02 - the discipline dealing with the art or science of applying scientific knowledge to practical problems
    - Engineering.n.03 - a room (as on a ship) in which the engine is located

There can also be cases, when two or more different synsets represent the same meaning. In that case they are synonyms and there is conceptual-semantic relation between the synsets. In WordNet they are called lemmas. An example of this is "engineering.n.02", "technology.n.02", engineering_science.n.01 and applied_science.n.01 that all have the same meaning: "the discipline dealing with the art or science of applying scientific knowledge to practical problems".

Today, there are more than 70 wordnets in the world proximately in 50 languages. WordNet is a registered trademark, owned by Princeton University. [12]

## 3.2 Natural Language Processing and Python NLTK

Natural language processing (NLP) is a field in artificial intelligence and computer sciences and is also related to the area of human-computer interaction. It tries to make computers interact with human language. Modern NLP algorithms are based on machine learning. Major tasks in NLP include automatic summarization to reduce text documents and to create a summary, machine translation, natural language understanding, part-of-speech tagging, speech recognition and question answering.

NLP is also being used to improve search engines. As opposed to regular well-known keyword search, SWSNL focuses on the meaning of the words and human natural language query formation [13]. All though search engines already understand the main concepts of the words people use in the process of searching data, there is still room for improvement. For example questions like “What buses drive from Mustamäe to Õismäe in Tallinn?” cannot be answered without natural language processing, because some vital words like “from” and “to” are meaningful for people, but not for computer if they do not have a concept. Part-of-speech tagging is also vital in these kinds of questions.

Python Natural Language Toolkit (NLTK) is a platform that allows Python programs to work with human language. It corporates nearly one hundred different built in trained modules. NLTK is ideally suited to students who are learning NLP or conducting research in NLP or closely related areas [14]. NLTK includes simple processing tasks such as tokenizing, parsing, stemming and chunking words. It is also possible to remove stop words and to do part-of-speech tagging.

## 3.3 Semantic Similarity and Semantic Relatedness

The most popular way for people to compare two objects or words and acquire knowledge about them, is to compare the similarity between those two objects. For humans, it is fairly easy to say if one word is more similar to a given word than another one, but computers have to have sorted system to do so [7]. Computers should also have some kind of expectation of what the word is about, like humans do. This is also known as the word-sense disambiguation (WSD) problem, which is an open problem in natural language processing and computational linguistics. WSD is identifying which meaning of a word is used in a sentence, when the word has various meanings.

Semantic similarity is an important topic in NLP and information and knowledge retrieval researches. It has also been used in cognitive science and artificial intelligence fields [7]. Semantic similarity is a metric defined so, that the distance between two concepts in a IS-A hierarchy tree also shows the similarity of two concepts. The higher the measure, the more related two concepts are. It is used to estimate the relationship between concepts and texts. If we take a word “Bicycle” then people can understand that the word is similar to “car”, because it is a sort of a vehicle.

The term semantic similarity is often confused with semantic relatedness. Semantic relatedness covers a broader range of relationships between concepts [15]. It also takes into account the definitions of both words. When before it was shown that bicycle is similar to car because of a IS-A relation, we can also state that bicycle is somewhat related to a helmet, road, and driving etc. This relation is called semantic relatedness.

## 3.4 WordNet Dictionary Based Semantic Similarity Measures

There are 4 types of semantic similarity measures based on WordNet. These include information content based measures, feature-based measure, path-based/edge counting measures and hybrid measures. [13]

Information content based measures takes into account the common information two synsets share. These include lowest common hypernyms of both synsets, length between two synsets and the length of the path from global root “entity” to synset [16]. For example, the lowest common hypernym for synsets “male person” and “female person” would be “person.” (Figure 1)

Feature based measure is similar to information content based measures in a way that it takes into account common properties both synsets have. The more common characteristics two concepts have and the less non-common characteristics they have, the more similar the concepts are [8]. Usually the definitions are used for finding common characteristics.

There are many proposals for measuring semantic similarity with path based measures. The main idea behind path-based measure is to create a function or a path that links two concept nodes together within a IS-A hierarchy [8]. This is reasonable way to measure

similarity because, a WordNet database is built in a tree-like hierarchy. All the following semantic similarity measures have a built in method in NLTK.

Hybrid measures are measures that include parts from all other semantic similarity measures and create a new one.

### 3.4.1 Definition Based Similarity

Definition based similarity is somewhat related to feature based similarity. This method uses the definitions of a word to understand the meaning of the synset. The relatedness to another synset is measured with the number of overlapping stemmed words in their dictionary definitions [16]. This method can only be used when the dictionary provides definitions of the words inside it. Similar idea was first introduced by Michael E. Lesk in 1986, also known as Lesk algorithm.

### 3.4.2 The Shortest Path Based Measure

The measure takes into account only the path length between two words [8]. It also has a fixed measure of max depth in taxonomy. It shows the depth of the IS-A hierarchy in a particular taxonomy. It is a constant and shows that the path length from the root node to the most distant leaf in any taxonomy is 16 [15]. The result of the shortest path based measure is in range from 0 to 1. The more similar the synsets are, the higher the result. $W_1$ and $W_2$ refer to comparable synsets.

$$sim_{path}(w_1, w_2) = 2 * max\_depth\_in\_taxonomy - length(w_1, w_2) \quad (1)$$

### 3.4.3 Wu & Palmer's Measure

Wu & Palmer's measure (WUP) takes into account the path length between two words, but also uses the most specific common concepts that links the two words. If we look Figure 1, then for the words "bus" and "train" the most specific common concept would be "public transport". The equation also includes depth, which is the length of the path to synset $W_i$ from the global root "entity" [8]. The result of the Wu & Pamer's measure is in range from 0 to 1. The more similar the synsts are, the higher the result.

$$sim_{WP}(w_1, w_2) = \frac{2*depth(common(w_1,w_2))}{length(w_1,w_2)+2*depth(common(w_1,w_2))} \quad (2)$$

### 3.4.4 Leacock & Chodorow's Measure

Leacock & Chodorow's measure (LCH) also takes into account the max depth in taxonomy measure. In this case this is also a fixed measure. Leacock-Chodorow is a logarithmic similarity measure and the measure is in range from 0 to 3.638. [8] [15]

$$sim_{LC}(w_1, w_2) = -log \frac{length(w_1, w_2)}{2 * max\_depth\_in\_taxonomy} \quad (3)$$

# 4 Experiments with Dictionary Based Similarities

The aim of this thesis is to propose a suitable way to find recommendable CETs using WordNet dictionary based semantic similarity methods. All the code examples starting from chapter 4.2 are done with an input query "engineering". If this were to be used by users, the query word would be selected by the users. To get the results and analyse the measures I also ran the codes through with 100 randomly selected words, which were selected from WordNet nouns using random generator, and 25 most used English words.

In this paragraph I first compare different path based semantic similarity measures and propose one hybrid unified method. I then analyse and create the definition based method to find suitable expansion terms. In the last sub-paragraph I am going to propose a hybrid method of both unified semantic similarity measure and definition based similarity.

## 4.1 Path Based Semantic Similarity Measures

There are many ways to measure semantic relations between words. In this sub-paragraph I am going to use edge-counting based similarity measures. These include the Shortest Path based measure (PATH), Wu and Palmer's measure (WUP) and Leacock-Chadorow's Similarity measure (LCH).

To compare the three different path based semantic similarity measures I selected a group of words from different taxonomies. In this part of the experiment I used only the first senses of the words. e.g. bicycle.n.01, boat.n.01, wheel.n.01, car.n.01, cat.n.01, word.n.01, entity.n.01, lion.n.01, love.n.01. This means that e.g. "boat" can also be part of other synsets and have meanings with other indexes like "boat.n.02" and "boat.n.03" etc. The first sense was chosen for simplicity reasons and to get an overview of the accuracy of the measures.

The program takes the first senses of both synsets and finds the result of similarity measures using the algorithms described in theoretical background paragraph. The three used similarity measures have built in methods in the Python NLTK package. All the

words are put through different semantic similarity measures separately and the results are presented in Table 1. Full code can be seen in Appendix 1.

Table 1 Comparison of path based similarities

| | **PATH** | **WUP** | **LCH** |
|---|---|---|---|
| ***Bicycle : Boat*** | 0.1667 | 0.7619 | 1.8458 |
| ***Bicycle : Wheel*** | 0.1111 | 0.6 | 1.4404 |
| ***Bicycle : Car*** | 0.2 | 0.7273 | 2.0281 |
| ***Bicycle : Cat*** | 0.0625 | 0.3478 | 0.8650 |
| ***Bicycle : Word*** | 0.0714 | 0.1333 | 0.9985 |
| ***Bicycle : Entity*** | 0.1111 | 0.2 | 1.4404 |
| ***Bicycle : Lion*** | 0.0588 | 0.3333 | 0.8044 |
| ***Bicycle : Love*** | 0.0666 | 0.125 | 0.9295 |

In Table 1 it can be seen that LCH measure has higher results than other measures. As mentioned before, the maximum result for Leacock-Chodorow's measure is 3.368. To compare Leacock-Chodorow measure equally with other measures I am going to transform the result into percentage so that the measure would be in range of 0 and 1.

$$transformed_{LCH} = \frac{result_{LCH}}{max_{LCH}} \quad (4)$$

Table 2 Comparison of path based similarities with transformed LCH measure

| | **PATH** | **WUP** | **LCH** |
|---|---|---|---|
| ***Bicycle : Boat*** | 0.1667 | 0.7619 | **0.5074** |
| ***Bicycle : Wheel*** | 0.1111 | 0.6 | **0.3960** |
| ***Bicycle : Car*** | 0.2 | 0.7273 | **0.5576** |
| ***Bicycle : Cat*** | 0.0625 | 0.3478 | **0.2378** |
| ***Bicycle : Word*** | 0.0714 | 0.1333 | **0.2745** |
| ***Bicycle : Entity*** | 0.1111 | 0.2 | **0.3960** |
| ***Bicycle : Lion*** | 0.0588 | 0.3333 | **0.2211** |
| ***Bicycle : Love*** | 0.0667 | 0.125 | **0.2555** |

In the Table 2 different semantic similarity measures are comparable since the relations between synsets are proportional. These similarity measures show relatively different results. First three examples are much more semantically similar than the comparison of the words “bicycle” and “cat” or “bicycle” and “love”. That is also understandable, taking into account that first three are to some extend related to vehicles and moving objects.

When comparing two words that belong to a different taxonomy, the results make less sense. Last five examples in Table 2 {cat, word, entity, lion, love} are selected specially to be from different taxonomies, which do not seem to have any similarity to a bicycle. This also explains why “love” and “boat” have very large difference in the similarity to “bicycle”. The words are far from each other in a IS-A hierarchy tree.

When looking to the results separately, it is clearly visible that the results of the Shortest Path based measures are underrated to a human mind. Humans are capable of generalization and abstract thinking, so it would make more sense, if we considered {bicycle, boat, wheel, car} more related to each other (Table 2). The fact that the Shortest Path based similarity results are small, could play a significant role in the process of finding suitable words for query expansion. It is possible to consider shortest path based similarity a good baseline for finding synsets that are not related to the initial word.

What can also be seen, is that the results of Leacock-Chodorow’s measure are fairly high with words that are not related to each other. This is particularly visible when we compere two words that are definitely not from the same taxonomy, e.g. {Bicycle : Love} and {Bicycle : Word}. The results for the same pairs with other measures are lower and represent the relationship between these words better.

The most unstable similarity measure is Wu & Palmer’s. For the relations that seem reasonable, it gives higher results than other measures, e.g. {Bicycle : Boat}, {Bicycle : Car} and {Bicycle : Wheel}. But for the relations that should be less relevant like {Bicycle : Cat} and {Bicycle : Entity} the result varies from higher than other similarity measures to lower than the results of other similarity measures. Although this measure is the most unstable, it gives the best results in comparison to other used similarity measures.

## 4.2 The Unified Semantic Similarity Measure

The need for a unified measure comes out when comparing the similarity measures separately. The Shortest Path based measure has relatively low similarity results, even with words that are similar. Leacock-Chodorow's results, on the other hand, are higher with the words that are not related to the query word. The results of Wu & Palmer's method vary from reasonable to not so reasonable, depending on the words compared.

To create a unified similarity measures I take all three previously mentioned similarity measures and combine them into one unified similarity measure. I considered two ways for creating a unified measure. One is a simple arithmetic average of the results of the methods, and other is a weighted arithmetic mean.

Using arithmetic average balances out the underrated result of Shortest Path based measure and overrated Leacock-Chodorow's similarity measure. (Equation 5)

$$new\_similarity = ((path + wup + lch)/3) \quad (5)$$

For using weighted arithmetic average, I experimented with various weights and decided to create a decision tree model that can be seen in Figure 2.

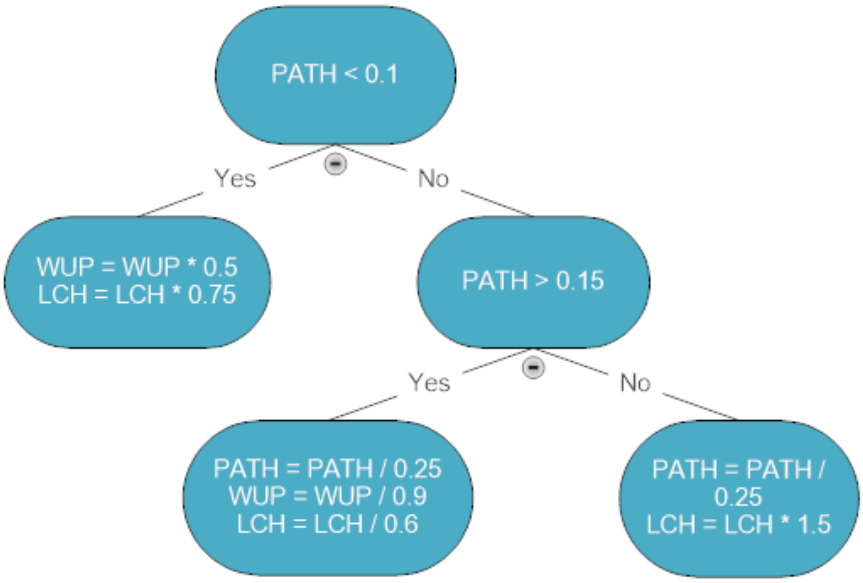


Figure 2 Decision tree for unified semantic similarity measures

If any of the results after this process were higher than 1, I made them equal to 1. This is necessary because the maximum results for measures individually was equal to 1.

From the table 3 it is visible, that the results for the weighted arithmetic measure seem to be more accurate than just using arithmetic average. In my following work I will be using the weighted arithmetic mean for getting the results.

Table 3 Difference of results of arithmetic mean and weighted arithmetic mean

| | **Arithmetic mean** | **Weighted arithmetic mean** |
|---|---|---|
| ***Bicycle : Boat*** | 0.4787 | 0.7863 |
| ***Bicycle : Wheel*** | 0.3690 | 0.5461 |
| ***Bicycle : Car*** | 0.4949 | 0.8458 |
| ***Bicycle : Cat*** | 0.2160 | 0.1383 |
| ***Bicycle : Word*** | 0.1598 | 0.1147 |
| ***Bicycle : Entity*** | 0.2357 | 0.4128 |
| ***Bicycle : Lion*** | 0.2044 | 0.1304 |
| ***Bicycle : Love*** | 0.1491 | 0.1069 |

After transforming LCH measure and creating new unified measure I can run in through the noun synsets of WordNet. The example is done with a word “engineering”. Full code can be seen in Appendix 2. To find similar words for the query using unified semantic similarity measure, the system has to:

1. Find all synsets that include query word.
2. Make sure that each synset in the user input query word is a noun.
3. Run through all the noun synsets in the WordNet database in pairs with all synsets that were found in section 1.
4. Find Shortest Path based, Wu & Palmer’s and Leacock & Chodorow’s semantic similarity measures separately for each of the pairs of synsets. Use the results of previous similarity measures to calculate the unified measure for each of the synset pairs.

```
def unified_measure(synset, word):
  path = synset.path_similarity(wordnet.synset(word))
  wup = synset.wup_similarity(wordnet.synset(word))
  lch = (synset.lch_similarity(wordnet.synset(word))) /
3.63758615973

  if path < 0.1:
    wup = wup * 0.5
    lch = lch * 0.75
  elif (path < 0.15 and path > 0.1):
    path = path / 0.25
    lch = lch * 1.5
  else:
    path = path / 0.25
    wup = wup / 0.9
    lch = lch / 0.6

  if path > 1:
    path = 1
  if wup > 1:
    wup = 1
  if lch > 1:
    lch = 1

  new_similarity = ((path + wup + lch) / 3)
  return new_similarity
```

5. If the result of the unified measure is higher than 0.6, the system will add the word to a dictionary of related synsets and print it out. The decision to choose 0.6 as the minimum limit was because results less than that did not give similar or related enough results to the query word. This can be seen in the WUP measure in Table 2. This was also examined with various other results, but the most reasonable results occurred with the minimum result of 0.6.

```
def similar(user_words):
 max_similarity = {}
 for synset in user_words:
  if synset in wordnet.all_synsets("n"):
   for word in read_synsets():
    result = unified_measure(synset, word)
    if result > 0.6:
     print ("Word: ", word, "Similarity: ", result)
     max_similarity[word] = result
 return max_similarity
```

## 4.3 Semantic Similarity Based on Synset Definition

In order to use definition based similarity for recommending related CETs, the definitions need to be pre-processed. The full code can be seen in appendix 3. Pre-processing the definition includes methods from natural language processing, such as tokenizing, removing stop words and stemming:

1. Getting the definition of a synset and tokenizing the definition:

```
# Tokenizing separates the definition into separate words
def tokenize(user_input):
 words = word_tokenize(user_input.definition())
 return words
```

2. Eliminating stop words from tokenized definition. Full list of WordNet stop words is in appendix 4. Removing stop words from the array, insures that there are no meaningless words to compare :

```
# Removes unnecessary stopwords
def remove_stopwords(words):
 clean_words = []
 for word in words:
  if not word in stopwords.words("english"):
   clean_words.append(word)
 return clean_words
```

3. Stemming the array of words. Stemming the words means, that words that belong to the definition of a user query word are decreased to the root stem of the word:

```
# Stemming reduces the word into its word stem
def stemming(clean_words):
 stemmed_words = []
 for word in clean_words:
  stemmed_words.append(ps.stem(word))
 return stemmed_words
```

With stemming the words, there is a possibility that two completely different words have the same stem. This could initiate a slight error in the results, but with using the combined hybrid method, this can be overlooked.

To find all synsets that are related to the query word, the program has to find all existing synsets and pre-process the definition of those synsets. The result is added to a dictionary.

```
user_word = "engineering"
# Returns an array with all the synsets of the query word
user_synsets = wordnet.synsets(user_word)
print user_synsets

# Adds user synsets and definitions into a dictionary
user_synsets_def={}
for synset in user_synsets:
 user_synsets_def[synset] =
stemming(remove_stopwords(tokenize(synset)))
print user_synsets_def
```

The same process is done with all the other words in WordNet database. This dictionary is written once into a text file and read from there, so the system would run faster.

```
# adds all synsets and definitions into a dictionary
all_synsets_def = {}
for synset in wordnet.all_synsets("n"):
 all_synsets_def[synset] =
stemming(remove_stopwords(tokenize(synset)))
```

After cleaning the definitions of all synsets, it is possible to find related words. For that both dictionaries are put through a function called “compare” that returns synsets that are related to the initial query word.

```
def compare (user_synsets_def, all_synsets_def):
 similar_synsets = []
 for all_key in all_synsets_def:
  for user_key in user_synsets_def:
   same_words = []
   count = 0
   for word_all in all_synsets_def.get(all_key):
    for word_user in user_synsets_def.get(user_key):
     if word_user == word_all and word_user not in same_words and
word_user != ";" and word_user != "(" and word_user != ")" and
word_user != ",":
      count += 1
      same_words.append(word_all)
   if same_words.__len__()>=2:
    print (user_key , all_key, "Same words: ", same_words, count)
    if all_key not in similar_synsets:
     similar_synsets.append(all_key)
 return similar_synsets
```

Two synsets are considered to be semantically similar and related when two or more stemmed words from their definitions match. If the code only returned synsets that have strictly more than two common stemmed words, the array of results would include very few results, mostly synsets from the same synset group. If the code returned synsets that have less than two common stemmed words, the array of results would have mostly not related synsets in it and the list of results would be immensely long. The difference between the amounts of recommendable synsets can be seen in Appendix 5. It is interesting because many of the words that have less same stemmed words in their definitions, are still quite related in the sense of a synset. This shows, that selecting two or more similar words as a criteria, actually makes more sense to a human mind.

## 4.4 Combined Hybrid Method

The third method of finding recommendable synsets for the initial query, is putting both previous methods together. This means that all the noun synsets of WordNet run through both methods, unified semantic similarity measure and definition based similarity, one after another. With the combined hybrid method, the results should be more accurate than just using one of the previous methods separately.

To put the methods together the synsets first run through a definition based similarity and then semantic similarity. There is a possibility to put the combined hybrid method together other way around, but after some testing I understood that this way the program takes longer time to run. The reason for that is, that the result of semantic similarities between all possible pairs of nouns in WordNet is not saved into a separate fail and the program cannot just read the result in. What is more, the results for the hybrid method constructed both ways, are the same. This is visible in appendix 6 with an example word “engineering”. The full code for combined hybrid method can be seen in appendix 7.

To analyse the results, the combined method is tested with only 125 words, which include 100 randomly selected words from WordNet and 25 most common nouns in English language. This is because natural language processing must be supported with human analytical skills.

# 5 Results and Analysis

In this paragraph I present some of the most interesting results and analyse the difference of the methods. I compare the results of the methods and their accuracy and conclude the results. I also include ideas for future work.

## 5.1 Results of the Unified Semantic Similarity Measure

The unified semantic similarity measure gives a pretty good overview of the words that could be related to the initial query word. Three example words are brought out in the table 4. The results are in an ordered list, where the most related synsets are in the beginning, and least related in the end of the column and can also be seen in Table 5.

Table 4 Results of unified semantic similarity measure

This table shows results for three different words. ### shows that there are many answers in between the start and end of the list. Total shows the amount of recommended synsets for an example word.

| **Engineering** | **Recreation Room** | **Tiger** |
|---|---|---|
| aeronautical_engineering.n.01 | canteen.n.04 | bengal_tiger.n.01 |
| aeronautical_engineering.n.02 | family_room.n.01 | big_cat.n.01 |
| application.n.01 | recreation_room.n.01 | cheetah.n.01 |
| architectural_engineering.n.01 | room.n.01 | false_saber-toothed_tiger.n.01 |
| automation.n.01 | rumpus_room.n.01 | feline.n.01 |
| automotive_technology.n.01 | anechoic_chamber.n.01 | jaguar.n.01 |
| bionics.n.01 | anteroom.n.01 | leopard.n.02 |
| biotechnology.n.02 | back_room.n.01 | leopardess.n.01 |
| chemical_engineering.n.01 | ballroom.n.01 | liger.n.01 |
| chemical_engineering.n.02 | barroom.n.01 | lion.n.01 |
| civil_engineering.n.01 | bathroom.n.01 | lion_cub.n.01 |
| communications_technology.n.01 | bedroom.n.01 | lioness.n.01 |
| computer_science.n.01 | belfry.n.02 | lionet.n.01 |
| computer_technology.n.01 | billiard_room.n.01 | panther.n.02 |

| Engineering | Recreation Room | Tiger |
|---|---|---|
| digital_communications_technology.n.01 | boardroom.n.01 | saber-toothed_tiger.n.01 |
| discipline.n.01 | cardroom.n.01 | smiledon_californicus.n.01 |
| electrical_engineering.n.01 | cell.n.06 | snow_leopard.n.01 |
| engineering.n.02 | cell.n.07 | tiger.n.02 |
| engineering.n.03 | chamber.n.03 | tiger_cub.n.01 |
| high_technology.n.01 | checkroom.n.01 | tiglon.n.01 |
| ### | ### | ### |
| Total: 4561 | Total: 1674 | Total: 12098 |

What is interesting, is that when using weighted arithmetic mean in the method, the amount of recommendable results becomes very high. The last row of table 4, shows the total results for the words. Using only this method, makes analysing the relatedness of synsets immensely difficult, since there are too many of recommended synsets.

Because the unified similarity measure recommends words that are the closest in the taxonomy and in the hierarchy, the results in the top of the list are quite good. But the last elements from the results are relatively random. This can be seen in table 5.

Table 5 Least similar results for the unified semantic similarity measure

| Engineering | Recreation Room | Tiger |
|---|---|---|
| ### | ### | ### |
| undercut.n.03 | cookfire.n.01 | virginia_deer.n.01 |
| urology.n.01 | dry_kiln.n.01 | wapiti.n.01 |
| veterinary_medicine.n.01 | gas_oven.n.01 | water_chevrotain.n.01 |
| villain.n.02 | limekiln.n.01 | western_big-eared_bat.n.01 |
| virology.n.01 | mausoleum_at_halicarnasus.n.01 | white-tailed_jackrabbit.n.01 |
| volcanology.n.01 | muffle.n.01 | wild_ass.n.01 |
| waltz.n.03 | oast.n.01 | wild_horse.n.01 |
| wave_mechanics.n.01 | stassano_furnace.n.01 | wild_sheep.n.01 |
| zymology.n.01 | taj_mahal.n.01 | workhorse.n.02 |

The amount of recommendations the system outputs, varies very much between different words, as it is shown in the last row of the Table 4. This also means that there are many words not shown in the table and that there are many words that are not related sense wise to the original query. From the middle of the list until the end, the list includes related synsets like other scientific research areas, other types of predators and different rooms, but as it can see in Table 5, there are also synsets that do not make much sense.

This shows the first problem of natural language processing. The computer does not understand the meaning of the word. To recommend the right words, the computer should have a sense of the context in which the query word is used, not only the relation in a graph. This is also known as the word sense disambiguation problem.

There are some rare instances, when the system returns few synsets or the initial synsets. These are words that are very specific. An example of this can be “knife fight”, “seven-up” or “ring finger”. The more specific the word is or the less homonyms it has and the less recommendable results it has. The more general the word, the more results it gets.

## 5.2 Results of the Definition Based Similarity

Definition based similarity mostly gave less results than the unified similarity measure, but there were also examples of words, that had very high amount of results. This is because there are many cases when definition of a synset includes very common words that are used in many different definitions. Special instance of that is an example word “chickaree”, which was one of the randomly selected words. “Chickeree” has a definition: “far western United States counterpart of the red squirrel”, which also means that the stemmed definition of this word includes stems like “unit” and “state”. Because of that, all words that have United States in their definitions are automatically put into the category of similar or related words. The array of recommendations includes states of USA, animals, plants etc. This resulted with 2636 synsets that have 2 or more common words in their definition. Clearly this is not accurate enough to make recommendations to a user with related words. Same problem occurs with notions like “united states house of representatives” or with words that include “north America” in their definitions.

This shows the second problem I encountered during the process of this thesis. Although stop words are eliminated, sometimes countries and cities should also be removed from

the stemmed definition of a word as well, because they add meaningless results without being in the preferred context. When cities or countries are used as the query word, the results include mostly only other cities or countries. This can be seen when running the code for instance with “Cannes”, “Cadiz”, “Sioux city” and “Haiphong”.

Third problem that came out during the experiment was that some words have very short definitions. This makes it impossible to compare anything to them. The results with the query word “person” led me to an error what I hadn’t thought about before. Synset “person.n.01” has a definition: “a human being”. The definition after tokenizing, removing stop words and stemming is just “[human]”. This means that word “being” from the “person.n.01” definition is both in the arrays of noun synsets of WordNet and in the array of stop words of WordNet. In this case the word is treated as a stop word and not as a meaningful noun. Full list of stop words is in appendix 4. This is another perfect example of the problem, where computers do not understand the sense of the word. It is not understandable for a machine, without knowing the context, when is “being” used as a verb “be” and when it is used as a noun with a definition: “the state or fact of existing” or “a living thing that has (or can develop) the ability to act or function independently”. What is more, is that there can never be 2 or more common stemmed words when comparing a word with a synset “person.n.01”, because this synset has only one word in the stemmed definition. This also means that the most common meaning of the word “person” was not used for finding recommendable related CETs.

I also understood, that when the definition of a word is long, the amount of common words should be higher. The amount of common words should grow exponentially. This is visible with a word “English hawthorn”, which has stemmed definition with 23 words, also including “European”, “north” and “America”.

What also came out with the user input query word “person”, was that this word also applies to a grammatical category used in the classification of pronouns, possessive determiners, and verb forms according to whether they indicate the speaker, the addressee, or a third party. This is no surprise when analysing the word, but it shows that when a user chooses a query input word, there are many synonyms people forget to think about and do not expect to get results to.

I found forth problem when analysing the word “tiger”. With unified semantic similarity measure, one of the highest recommendations was “Bengal tiger”, but this recommendation did not come out with definition based similarity. The reason for that was, that word “tiger” does not have “tiger” in its definition, but “Bengal tiger” is defined with the word “tiger”. This led me to a conclusion, that the search word itself should also be counted as the common word in definitions.

## 5.3 Results of the Combined Hybrid Method

Combined hybrid method resulted with minimized amount of recommendable related words. All the recommendations made with this method are related to the initial query and provide useful and meaningful relations. The amount of words recommended is small enough for humans to overview it manually and to analyse the relatedness of each synset. Overall this method provides most related synsets by the means of meaning and closeness in taxonomy. This means that the method ensures that words are actually related to each other, not only include same words in their definitions. It combines the best attributes from both previous methods. There were still some words that got only 1-3 recommendable CETs. This means that the results were narrowed down too much. In that case, only unified similarity measure or definition based similarity should be used.

In case of query “engineering” the results were narrowed down from 4561 possible related words recommended with unified semantic similarity measure to only 61 possible synsets. When word “tiger” had 672 results with unified semantic similarity measure, and only 168 results with definition based similarity, the combined hybrid method resulted with 45 recommendations.

When the query word is a plant and has a long definition, even the combined hybrid method cannot narrow down the results. This was extremely visible with the query “English hawthorn”. This is because it has a stemmed definition of 23 words. It also includes words such as “European”, “North” and “America”. The problem what occurs with places is described in previous sub-paragraph. And because it is a plant, the taxonomy in which it is, is really wide. This means, that there are very many related synsets even when using the combined hybrid method. The query resulted with 3242 CETs, in which most of the words were other plants and biological terms. These kinds of problems can be solved when setting more criterias for definition based similarity.

Most of the problems that occurred with using other methods, did not play any significant role with finding words with this combined hybrid method. This is because the interim results were not exposed in the outcome. There were some instances when the system recommended some synsets that were related to the less common homonyms of the initial query word, but this is because of the first problem proposed in sub-paragraph "Results of the unified semantic similarity measure". The program did not take into account the context the word was presented and found all possible results, irrespective of the actual context the user would have wanted it. In case, where initial query is given without any context, the system outputs all possible results.

The combined hybrid method could be enhanced when the problems of previous methods are solved and the suggestions are put into use.

## 5.4 Results for the Experiments with the Most Used English Words

According to Oxford Dictionary, the 25 most common nouns of English language include time, year, people, way, day, man, thing, woman, life, child, world, school, state, family, student, group, country, problem, hand, part, place, case, week, company and system [17]. These are words that have many homonyms and also many possible meanings and alternatives of uses. That is the reason why they result with the highest amount of possible recommendations.

For example the word "time" is part of 15 different synsets: {time.n.01, time.n.02, time.n.03, time.n.04, time.n.05, time.n.06, clock_time.n.01, fourth_dimension.n.01, meter.n.04, prison_term.n.01, clock.v.01, time.v.02, time.v.03, time.v.04, time.v.05}. This word resulted with 298 recommendable synsets when using the combined hybrid method, but when only the unified semantic similarity measure was used, the number of recommendable results was 11080.

Compared to other words used for the experiments, the most commonly used words in English have the highest amount of recommendable CETs when using combined hybrid method, but the amount is still manageable for humans to analyse.

## 5.5 Conclusion and Future Work

The results of the experiments done in thesis brought out many key points that make finding related CETs and working with natural language processing complicated.

The main reflection points I encountered during the thesis:

1. The computer does not understand the meaning of the word. To recommend the right words, the computer should have a sense of the context in which the query word is used. Also known as the WSD problem.
2. There are some stop words that need to be considered both as nouns and stop words by their definition. An example of this is word “being”.
3. It is reasonable to eliminate countries and cities from the stemmed definitions, because they add meaningless results without being in the context. This of course depends of the search word.
4. Some words have very short definitions, which makes it impossible to compare anything to them using only definition based similarity.
5. Some words have very long definitions, which means that the amount of necessary common words should grow exponentially with the amount of words in the definition.
6. The initial query words should also be used as possible common words when using the definition based similarity, to ensure that “types of” relationships are also included in the outcome.

Overall there are pros and cons for each of the three proposed methods. With unified semantic similarity measure, the highest results include mostly synsets that are related to the main idea of the query word, but because of homonymy, there are also synsets that, at first look, seem unreasonable to recommend. This method also recommends words that are not related to the original query, but in the taxonomy, have the same distance as the related words. This method gave the highest amount of results compared to other methods in total.

When using definition based similarity, the outcome of the system included smaller amount of synsets than the previous method. The more the synsets had common words in their definitions, the more related the synsets were. But as with previous method, this method also encountered some problems. If the definition includes a location, many not

related words are automatically added to the related synset list. This method is also useless, when the stemmed definition is shorter than 3 words.

With the combined hybrid method, the amount of recommendable results was minimized. This is because definition based similarity chooses all possible words that could have a relatedness to the original query based on their definition and unified semantic similarity measure narrows it down to the synsets that belong to the same taxonomy. This is the method that I would recommend as the most useful one. The results include mostly related words, with one or two exceptions, depending of the amount of related synsets the system chose to output.

For future work I would insert the exceptions I encountered in this experiment to the comparing code and analyse the new results. I would also look into EstNLTK and Estonian WordNet to see, if the proposed methods would work with Estonian language. I would also improve on the programming side to make the methods work faster. When this system was public to the users, it could be enhanced with recommending systems and user ratings to take into account the human ability to analyse.

To conclude, it is difficult to judge the results of natural language processing without the help of a human understanding of the words. There are many key points that need to be thought through before programming the system based on dictionary based similarities. Depending of the aim of finding related words, all methods are useful, but to get the most related and minimized amount of results of related synsets, the combined hybrid method gives the most accurate results. The outcome of this thesis can be used for query expansion. It can also be used to create automatic mind-maps to improve the ideation phase of creative thinking.

# Summary

Internet users frequently use incomplete query terms or human language near sentence construction to search for data from search engines. This problem is usually solved with query expansion.

The aim of this was to find suitable way to find recommendable terms for query expansion using WordNet dictionary based similarities, and to find problems and key point to take into account, when working with dictionary based similarities and natural language processing. The recommended terms had to be related sense wise to the original query word used in the system, so if they were used in query expansion, they would provide the user with additional results and data. All stated objectives were met in this thesis.

In this thesis three methods were proposed for finding suitable terms for query expansion using WordNet dictionary based measures of similarity. The methods included unified path based measure, definition based measure and a combined hybrid method of two previous measures.

From the results it was visible that the best way for finding recommendable terms is using combined hybrid method. It resulted with minimized amount of recommendations and nearly all of the recommendations are related to the initial query word. It was also stated that for recommending related words, the computer should have some knowledge about the context were word is used. It also came out that when using definition based similarity, there are various difficulties and key points that play a significant role in recommending the words.

# References


[1] S. Jain, “Trends, Problems and Solutions of Recommender Systems,” in *Computing, Communication & Automation (ICCCA)*, Noida, 2015.

[2] P. André , J. Teevan and S. T. Dumais, “From X-Rays to Silly Putty via Uranus: Serendipity and its Role in Web Search,” in *SIGCHI Conference on Human Factors in Computing Systems*, New York, 2009.

[3] J. Zhang, B. Deng and X. Li, “Concept Based Query Expansion Using WordNet,” in *International e-Conference on Advanced Science and Technology*, Dajeon, 2009.

[4] “Princeton University "About WordNet.",” Princeton University, 2010. [Online]. Available: http://wordnet.princeton.edu.

[5] S. Bird, E. Klein and E. Loper, Natural Language Processing with Python, O'Reilly Media, Inc, 2009.

[6] S. K. Manjula, D. K. C. Shet and D. U. Acharya, “A New Similarity Measure for Taxonomy Based on Edge Counting,” *International Journal of Web & Semantic Technology (IJWesT)*, vol. 3, no. 4, 2012.

[7] M. G. Ahsaee , M. Naghibzadeh and S. E. Y. Naieni, “Weighted Semantic Similarity Assessment Using WordNet,” in *International Conference on Computer & Information Science (ICCIS)*, Kuala Lumpeu, 2012.

[8] L. Meng, R. Huang and J. Gu, “A Review of Semantic Similarity Measures in WordNet,” *International Journal of Hybrid Information Technology*, vol. 6, no. 1, 2013.

[9] D. Pal, M. Mitra and K. Datta, “Improving Query Expansion Using WordNet,” *Journal of the Association for Information Science & Technology*, vol. 65, no. 12, pp. 2469-2478, 2014.

[10] C. Fellbaum, WordNet - An Electronic Lexical Database, Cambridge: Massachusetts Institute of Technology, 1998.

[11] S. Patwardhan, S. Banerjee and T. Pedersen, “Using Measures of Semantic Relatedness for Word Sense Disambiguation,” *Proceedings of the 4th international conference on Computational linguistics and intelligent text processing*, pp. 241-257 , 2003.

[12] A. Lohk, A System of Test Patterns to Check and Validate the Semantic Hierarchies of WordNet-type Dictionaries, Tallinn: TTÜ Kirjastus, 2015.

[13] G. Varelas, E. Voutsakis, P. Raftopoulou, E. G. Petrakis and E. E. Milios, “Semantic Similarity Methods in WordNet and Their Application to Information Retrieval on the Web,” in *7th annual ACM international workshop on Web information and data management*, New York, 2005.

[14] I. Habernal and M. Konopik, “SWSNL: Semantic Web Search Using Natural Language,” *Expert Systems with Applications*, vol. 40, no. 9, pp. 3649-3664, 2013.

[15] S. Bird, “NLTK: The Natural Language Toolkit,” *Proceedings of the COLING/ACL on Interactive presentation sessions*, 2006.

[16] A. Budanitsky and G. Hirst, “Evaluating WordNet-based measures of lexical semantic relatedness,” *Computational Linguistics. ,* vol. 32, no. 1, pp. 13-47, 2006.

[17] “Oxforddictionaries.com,” Oxford Dictionaries, [Online]. Available: http://www.oxforddictionaries.com/words/what-can-corpus-tell-us-about-language. [Accessed 4 May 2016].

# Appendix 1 – Examples of Semantic Similarity Measures

```
from nltk.corpus import wordnet
word_one = wordnet.synset("bicycle.n.01")
word_two = wordnet.synset("boat.n.01")
word_three = wordnet.synset("wheel.n.01")
word_four = wordnet.synset("car.n.01")
word_five = wordnet.synset("cat.n.01")
word_six = wordnet.synset("word.n.01")
word_seven = wordnet.synset("entity.n.01")
word_eight = wordnet.synset("lion.n.01")
word_nine = wordnet.synset("love.n.01")

print("Path bicycle vs boat:
"+str(word_one.path_similarity(word_two)))
print("Path bicycle vs wheel:
"+str(word_one.path_similarity(word_three)))
print("Path bicycle vs car:
"+str(word_one.path_similarity(word_four)))
print("Path bicycle vs cat:
"+str(word_one.path_similarity(word_five)))
print("Path bicycle vs word:
"+str(word_one.path_similarity(word_six)))
print("Path bicycle vs entity:
"+str(word_one.path_similarity(word_seven)))
print("Path bicycle vs lion:
"+str(word_one.path_similarity(word_eight)))
print("Path bicycle vs love:
"+str(word_one.path_similarity(word_nine)))

print("WUP bicycle vs boat: " +
str(word_one.wup_similarity(word_two)))
print("WUP bicycle vs wheel: " +
str(word_one.wup_similarity(word_three)))
print("WUP bicycle vs car: " +
str(word_one.wup_similarity(word_four)))
print("WUP bicycle vs cat: " +
str(word_one.wup_similarity(word_five)))
print("WUP bicycle vs word: "+str(word_one.wup_similarity(word_six)))
print("WUP bicycle vs entity:
"+str(word_one.wup_similarity(word_seven)))
print("WUP bicycle vs lion:
"+str(word_one.wup_similarity(word_eight)))
print("WUP bicycle vs love: "+str(word_one.wup_similarity(word_nine)))
```

```
print("LCH bicycle vs boat: "+str(word_one.lch_similarity(word_two)))
print("LCH bicycle vs wheel:
"+str(word_one.lch_similarity(word_three)))
print("LCH bicycle vs car: "+str(word_one.lch_similarity(word_four)))
print("LCH bicycle vs cat: "+str(word_one.lch_similarity(word_five)))
print("LCH bicycle vs word: "+str(word_one.lch_similarity(word_six)))
print("LCH bicycle vs entity:
"+str(word_one.lch_similarity(word_seven)))
print("LCH bicycle vs lion:
"+str(word_one.lch_similarity(word_eight)))
print("LCH bicycle vs love: "+str(word_one.lch_similarity(word_nine)))

print("New LCH bicycle vs boat:
"+str(word_one.lch_similarity(word_two)/3.63758615973))
print("New LCH bicycle vs wheel:
"+str(word_one.lch_similarity(word_three)/3.63758615973))
print("New LCH bicycle vs car:
"+str(word_one.lch_similarity(word_four)/3.63758615973))
print("New LCH bicycle vs cat:
"+str(word_one.lch_similarity(word_five)/3.63758615973))
print("New LCH bicycle vs word:
"+str(word_one.lch_similarity(word_six)/3.63758615973))
print("New LCH bicycle vs entity:
"+str(word_one.lch_similarity(word_seven)/3.63758615973))
print("New LCH bicycle vs lion:
"+str(word_one.lch_similarity(word_eight)/3.63758615973))
print("New LCH bicycle vs love:
"+str(word_one.lch_similarity(word_nine)/3.63758615973))
```

# Appendix 2 – Full Code of Unified Semantic Similarity

```
import json
from nltk.corpus import wordnet
from nltk.corpus import wordnet_ic

def read_synsets():
 file = open("all_synsets.txt", "r+")
 string = file.read()
 all_synsets_def = json.loads(string)
 file.close()
 return all_synsets_def

def unified_measure(synset, word):
 path = synset.path_similarity(wordnet.synset(word))
 wup = synset.wup_similarity(wordnet.synset(word))
 lch = (synset.lch_similarity(wordnet.synset(word))) / 3.63758615973

 if path < 0.1:
  wup = wup * 0.5
  lch = lch * 0.75
 elif (path < 0.15 and path > 0.1):
  path = path / 0.25
  lch = lch * 1.5
 else:
  path = path / 0.25
  wup = wup / 0.9
  lch = lch / 0.6
 if path > 1:
  path = 1
 if wup > 1:
  wup = 1
 if lch > 1:
  lch = 1
 new_similarity = ((path + wup + lch) / 3)
 return new_similarity

def similar(user_words):
 max_similarity = {}
 for synset in user_words:
  if synset in wordnet.all_synsets("n"):
   for word in read_synsets():
    result = unified_measure(synset, word)
    if result > 0.6:
     print ("Word: ", word, "Similarity: ", result)
     max_similarity[word] = result
 return max_similarity

word = "engineering"
user_words = wordnet.synsets(word)
print user_words
similar(user_words)
```

# Appendix 3 – Full Code of Definition Based Similarity

```
import json
from nltk.corpus import wordnet, stopwords
from nltk.stem import PorterStemmer
from nltk.tokenize import word_tokenize

# reads in already pre-processed synset definitions from a file
def read_synsets():
 file = open("all_synsets.txt", "r+")
 string = file.read()
 all_synsets_def = json.loads(string)
 file.close()
 return all_synsets_def

# Tokenizing separates the definition into separate words
def tokenize(user_input):
 words = word_tokenize(user_input.definition())
 return words

# Removes unnecessary stopwords
def remove_stopwords(words):
 clean_words = []
 for word in words:
  if not word in stopwords.words("english"):
   clean_words.append(word)
 return clean_words

# Stemming reduces the word into its word stem
def stemming(clean_words):
 stemmed_words = []
 for word in clean_words:
  stemmed_words.append(ps.stem(word))
 return stemmed_words
```

```
def compare (user_synsets_def, all_synsets_def):
 similar_synsets = []
 for all_key in all_synsets_def:
  for user_key in user_synsets_def:
   same_words = []
   count = 0
   for word_all in all_synsets_def.get(all_key):
    for word_user in user_synsets_def.get(user_key):
     if word_user == word_all and word_user not in same_words and
word_user != ";" and word_user != "(" and word_user != ")" and
word_user != ",":
      count += 1
      same_words.append(word_all)
   if same_words.__len__()>=2:
    print (user_key , all_key, "Same words: ", same_words, count)
    if all_key not in similar_synsets:
     similar_synsets.append(all_key)
 return similar_synsets

ps = PorterStemmer()
user_word = "engineering"
# Returns an array with all the synsets of the query word
user_synsets = wordnet.synsets(user_word)
print user_synsets
# Adds user synsets and definitions into a dictionary
user_synsets_def={}
for synset in user_synsets:
 user_synsets_def[synset] =
stemming(remove_stopwords(tokenize(synset)))
print user_synsets_def

compare(user_synsets_def, read_synsets())
```

# Appendix 4 – Stemmed Stop Words of WordNet

Table 6

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| i | herself | are | if | above | where | so | ain |
| me | it | was | or | below | why | than | aren |
| my | its | were | because | to | how | too | couldn |
| myself | itself | be | as | from | all | very | didn |
| we | they | been | until | up | any | s | doesn |
| our | them | being | while | down | both | t | hadn |
| ours | their | have | of | in | each | can | hasn |
| ourselves | theirs | has | at | out | few | will | haven |
| you | themselves | had | by | on | more | just | isn |
| your | what | having | for | off | most | don | ma |
| yours | which | do | with | over | other | should | mightn |
| yourself | who | does | about | under | some | now | mustn |
| yourselves | whom | did | against | again | such | d | needn |
| he | this | doing | between | further | no | ll | shan |
| him | that | a | into | then | nor | m | shouldn |
| his | these | an | through | once | not | o | wasn |
| himself | those | the | during | here | only | re | weren |
| she | am | and | before | there | own | ve | won |
| her | is | but | after | when | same | y | wouldn |
| hers | | | | | | | |

# Appendix 5 – Comparison of the Results of Definition Based Similarity

Table 7 Results for definition based similarity when common words is 2 or more

| | | |
|---|---|---|
| engineer.n.01 | automotive_engineer.n.01 | anesthesiology.n.01 |
| artificial_intelligence.n.01 | landscape.n.03 | mechanical_engineering.n.01 |
| biomedicine.n.01 | theorist.n.01 | information_theory.n.01 |
| applied_psychology.n.01 | aristotelianism.n.01 | harley_street.n.01 |
| bionics.n.01 | propaedeutic.n.01 | management_personnel.n.01 |
| fortification.n.02 | muse.n.01 | gibraltar.n.01 |
| pragmatism.n.01 | agriculturist.n.01 | homing_torpedo.n.01 |
| steamer.n.03 | gastronomy.n.02 | teflon.n.01 |
| mercantile_law.n.01 | foreground.n.02 | philanthropic_foundation.n.01 |
| painting.n.01 | wisdom.n.03 | strategy.n.02 |
| binary_file.n.01 | braun.n.02 | chemistry.n.01 |
| station.n.03 | master_of_arts.n.01 | aesthetics.n.01 |
| book_of_knowledge.n.01 | military_science.n.01 | meteorology.n.02 |
| organon.n.01 | project.n.02 | abstainer.n.01 |
| cooper_union.n.01 | medical_science.n.01 | serology.n.01 |
| interphone.n.01 | empiricism.n.02 | science.n.01 |
| civil_engineer.n.01 | aeronautical_engineering.n.01 | ontology.n.01 |
| studio.n.01 | resource.n.03 | sickbay.n.01 |
| mathematics.n.01 | monkey_bridge.n.01 | electrical_engineer.n.01 |
| probability_theory.n.01 | associate_in_applied_science.n.01 | partition.n.02 |
| agronomy.n.01 | cabin.n.01 | bachelor_of_arts.n.01 |
| life_science.n.01 | job.n.11 | aeronautical_engineer.n.01 |
| technology.n.01 | bell_book.n.01 | neurology.n.01 |
| profession.n.02 | approach.n.01 | clinical_anatomy.n.01 |
| architecture.n.02 | biomedical_science.n.01 | radiology.n.01 |
| self-discipline.n.01 | civil_engineering.n.01 | chemical_engineering.n.01 |
| biophysics.n.01 | disambiguator.n.01 | postmodernism.n.01 |

| | | |
|---|---|---|
| intercommunication_system.n.01 | roebling.n.01 | department_of_defense_laboratory_system.n.01 |
| epidemiology.n.01 | license.n.02 | technical_school.n.01 |
| architectural_engineering.n.01 | cooper.n.01 | academic_program.n.01 |
| yoga.n.02 | army_high_performance_computing_research_center.n.01 | geriatrics.n.01 |
| yoga.n.01 | oceanography.n.01 | steinman.n.01 |
| diderot.n.01 | junkers.n.01 | academy.n.02 |
| auxiliary_engine.n.01 | national_institutes_of_health.n.01 | unwieldiness.n.01 |
| elegance.n.02 | martial_art.n.01 | natural_language_processing.n.01 |
| tactics.n.01 | problem_solver.n.01 | tribology.n.01 |
| technique.n.01 | quackery.n.01 | medicine.n.01 |
| geology.n.01 | strategics.n.01 | marine_engineer.n.01 |
| naval_engineering.n.01 | pragmatist.n.02 | economics.n.01 |
| naval_research_laboratory.n.01 | physiology.n.01 | chemical_engineering.n.02 |
| polytechnic_institute.n.01 | versification.n.03 | valve-in-head_engine.n.01 |
| forensic_medicine.n.01 | scientist.n.01 | back_room.n.01 |
| shipping_room.n.01 | earth_science.n.01 | bush.n.05 |
| encyclopedia.n.01 | highway_engineer.n.01 | engineering.n.02 |
| nuclear_engineering.n.01 | theory.n.01 | nosology.n.01 |
| pharmacy.n.01 | text_editor.n.02 | institute.n.01 |
| rocketry.n.01 | wells.n.01 | engineering.n.03 |

Table 8 Results for definition based similarity when common words is more than 2

| | | |
|---|---|---|
| engineer.n.01 | engineering.n.02 | technique.n.01 |
| artificial_intelligence.n.01 | aesthetics.n.01 | naval_research_laboratory.n.01 |
| biomedicine.n.01 | technical_school.n.01 | polytechnic_institute.n.01 |
| applied_psychology.n.01 | geriatrics.n.01 | engineering.n.03 |
| technology.n.01 | chemical_engineering.n.02 | |

# Appendix 6 – Comparison of Combined Hybrid Method Results

| Unified -> Definition | Definition -> Unified |
|---|---|
| aeronautical_engineering.n.01 | tribology.n.01 |
| architectural_engineering.n.01 | aesthetics.n.01 |
| bionics.n.01 | applied_psychology.n.01 |
| chemical_engineering.n.01 | bionics.n.01 |
| chemical_engineering.n.02 | chemical_engineering.n.01 |
| civil_engineering.n.01 | architectural_engineering.n.01 |
| engineering.n.02 | chemistry.n.01 |
| engineering.n.03 | mathematics.n.01 |
| mechanical_engineering.n.01 | artificial_intelligence.n.01 |
| naval_engineering.n.01 | cooper_union.n.01 |
| nuclear_engineering.n.01 | economics.n.01 |
| profession.n.02 | profession.n.02 |
| rocketry.n.01 | propaedeutic.n.01 |
| technology.n.01 | serology.n.01 |
| artificial_intelligence.n.01 | sickbay.n.01 |
| back_room.n.01 | tactics.n.01 |
| shipping_room.n.01 | back_room.n.01 |
| sickbay.n.01 | biomedicine.n.01 |
| tribology.n.01 | civil_engineering.n.01 |
| architecture.n.02 | engineering.n.02 |
| military_science.n.01 | geriatrics.n.01 |
| science.n.01 | martial_art.n.01 |
| cabin.n.01 | meteorology.n.02 |
| agronomy.n.01 | quackery.n.01 |
| mathematics.n.01 | radiology.n.01 |
| strategics.n.01 | strategy.n.02 |
| strategy.n.02 | agronomy.n.01 |
| tactics.n.01 | architecture.n.02 |
| aesthetics.n.01 | forensic_medicine.n.01 |

| Unified -> Definition | Definition -> Unified |
|---|---|
| applied_psychology.n.01 | life_science.n.01 |
| chemistry.n.01 | nuclear_engineering.n.01 |
| earth_science.n.01 | aeronautical_engineering.n.01 |
| economics.n.01 | epidemiology.n.01 |
| life_science.n.01 | gastronomy.n.02 |
| natural_language_processing.n.01 | mechanical_engineering.n.01 |
| biomedical_science.n.01 | medical_science.n.01 |
| biophysics.n.01 | natural_language_processing.n.01 |
| geology.n.01 | naval_engineering.n.01 |
| medical_science.n.01 | oceanography.n.01 |
| meteorology.n.02 | pharmacy.n.01 |
| oceanography.n.01 | rocketry.n.01 |
| probability_theory.n.01 | anesthesiology.n.01 |
| quackery.n.01 | cabin.n.01 |
| medicine.n.01 | chemical_engineering.n.02 |
| neurology.n.01 | engineering.n.03 |
| physiology.n.01 | geology.n.01 |
| radiology.n.01 | medicine.n.01 |
| serology.n.01 | neurology.n.01 |
| cooper_union.n.01 | nosology.n.01 |
| empiricism.n.02 | physiology.n.01 |
| gastronomy.n.02 | biomedical_science.n.01 |
| martial_art.n.01 | empiricism.n.02 |
| propaedeutic.n.01 | probability_theory.n.01 |
| project.n.02 | technology.n.01 |
| anesthesiology.n.01 | biophysics.n.01 |
| biomedicine.n.01 | earth_science.n.01 |
| epidemiology.n.01 | military_science.n.01 |
| forensic_medicine.n.01 | science.n.01 |
| geriatrics.n.01 | shipping_room.n.01 |
| nosology.n.01 | strategics.n.01 |
| pharmacy.n.01 | project.n.02 |

# Appendix 7 – Full Code of Combined Hybrid Method

```
import json
from nltk.corpus import wordnet, stopwords
from nltk.stem import PorterStemmer
from nltk.tokenize import word_tokenize

def tokenize(user_input):
 words = word_tokenize(user_input.definition())
 return words

def remove_stopwords(words):
 clean_words = []
 for word in words:
  if not word in stopwords.words("english"):
   clean_words.append(word)
 return clean_words

def stemming(clean_words):
 stemmed_words = []
 for word in clean_words:
  stemmed_words.append(ps.stem(word))
 return stemmed_words

def read_synsets():
 file = open("all_synsets.txt", "r+")
 string = file.read()
 all_synsets_def = json.loads(string)
 file.close()
 return all_synsets_def

def compare (user_synsets_def, all_synsets_def):
 similar_synsets = {}
 for all_key in all_synsets_def:
  for user_key in user_synsets_def:
   same_words = []
   count = 0
   for word_all in all_synsets_def.get(all_key):
    for word_user in user_synsets_def.get(user_key):
     if word_user==word_all and word_user not in same_words and
word_user != ";" and word_user != "(" and word_user != ")" and
word_user != ",":
      count += 1
      same_words.append(word_all)
   if same_words.__len__()>=2:
    print (all_key, "Same words: ", same_words, count)
    if all_key not in similar_synsets:
     similar_synsets[all_key] = count
 return similar_synsets
```

```
def new_measure_change_path(word, synset):
 path = synset.path_similarity(wordnet.synset(word))
 wup = synset.wup_similarity(wordnet.synset(word))
 lch = (synset.lch_similarity(wordnet.synset(word))) / 3.63758615973
 new_similarity = (path + wup + lch)/3
 return new_similarity

def similarity_measures(similar_synsets, user_synsets_def):
 max_similarity = {} #tyhi dictionary
 for synset in user_synsets_def:
  if synset in wordnet.all_synsets("n"):
   for word in similar_synsets: #key in dictionary
    result = new_measure_change_path(word, synset)
    if result > 0.6:
     print (word, "Similarity: ", result)
     max_similarity[synset] = result
 return max_similarity

def process(user_word):
 user_synsets = wordnet.synsets(user_word)
 print user_synsets
 user_synsets_def={}
 for synset in user_synsets:
  user_synsets_def[synset] =
stemming(remove_stopwords(tokenize(synset)))

 similarity_measures(compare(user_synsets_def, read_synsets()),
user_synsets_def)
 print "\n"

ps = PorterStemmer()

user_word = "engineering"
process(user_word)
```