TALLINN UNIVERSITY OF TECHNOLOGY
School of Information Technologies

Renet Rämman 211477IAPM

GENERATING PROBABILISTIC CLASSIFICATION RULES FROM EXISTING KNOWLEDGE BASES

Master's thesis

Supervisor: Tanel Tammet
Ph.D

Tallinn 2023

TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Renet Rämman 211477IAPM

OLEMASOLEVATE TEADMUSBAASIDE PEALT UUTE TÕENÄOSUSLIKE KLASSIKUULUVUSREEGLITE JÄRELDAMINE

Magistritöö

Juhendaja: Tanel Tammet
Ph.D

Tallinn 2023

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: Renet Rämman

07.05.2023

Abstract

This thesis is about developing a method for generating probabilistic classification rules from existing knowledge bases and using the developed method to create a knowledge base of probabilistic classification rules. Multiple methods exist for generating rules from different sources, however probabilistic classification rules can not be found from well known knowledge bases. Probabilistic classification rules are rules which can be used to infer probabilistic classification such as "If an animal has a trunk, that animal is likely an elephant.". This probabilistic classification can be described by the following rule *0.9: X|hasA|trunk => X|isA|elephant*. Supposability values are probability-like numbers prepended to the classification rules which can used for comparing the relative probabilities of the rules in relation to each other.

The method developed for this thesis is able to generate new classification rules from existing knowledge bases, and calculate supposability values for those rules. Such probabilistic classification rules can be used for commonsense reasoning among other tasks.

A knowledge base of 1973 probabilistic classification rules was created using this method. A subset of the created knowledge base was manually evaluated.

This thesis is written in English and is 52 pages long, including 5 chapters, 13 figures and 4 tables.

Annotatsioon

Olemasolevate teadmusbaaside pealt uute tõenäosuslike klassikuuluvusreeglite järeldamine

Antud töö käsitleb tõenäosuslike klassikuuluvusreeglite genereerimist olemasolevatest teadmusbaasidest. Töö käigus arendatud meetodit kasutatakse tõenäosusliku klassikuuluvuse teadmusbaasi loomiseks. On mitmeid meetodeid reeglite genereerimiseks erinevatest allikatest, kuid tõenäosuslike klassikuuluvusreegleid tuntud teadmistebaasidest ei leita. Tõenäosuslikud klassikuuluvusreeglid on reeglid, millest saab tuletada tõenäosusliku klassikuuluvust, nagu näiteks „Kui loomal on lont, siis ta on suure tõenäosusega elevant.”. Seda tõenäosusliku klassikuuluvust väljendab järgmine reegel: *0.9: X|omab|lont => X|on|elevant*. Tõenäosusväärtused on tõenäosuse-sarnased numbrid, mis on lisatud tõenäosuslike klassikuuluvusreeglite ette. Neid numbreid saab kasutada reeglite tõenäosuste omavaheliseks võrdlemiseks.

Selle töö jaoks väljatöötatud meetod suudab olemasolevate teadmustebaaside pealt genereerida uusi klassikuuluvusreegleid ja arvutada nendele reeglitele tõenäosusväärtused. Selliseid tõenäosuslikke klassikuuluvusreegleid saab kasutada muu hulgas tavateadmiste argumenteerimiseks.

Töö käigus loodi 1973 tõenäosusliku klassifitseerimisreegliga teadmustebaas. Osa loodud teadmusbaasist hinnati manuaalselt.

Lõputöö on kirjutatud inglise keeles ning sisaldab teksti 52 leheküljel, 5 peatükki, 13 joonist, 4 tabelit.

List of abbreviations and terms

LLM	Large language model
AI	Artificial Intelligence
MGI	Magic Gini Impurity
NLP	Natural language processing
NLTK	Natural language toolkit [17]
PCA	Principal component analysis
ISC	Improved sqrt-cosine

Table of Contents

List of Figures

List of Tables

1 Introduction

Knowledge bases are repositories of information. They contain rules and facts that a typical human or an expert in a given field would consider as common sense, or that are relevant to a particular field or organization. A useful trait of knowledge bases is that they are made to be machine-readable. This makes them easy to use and useful for machine learning applications, search engines virtual assistants, etc. Overall, a knowledge base serves as a valuable resource for intelligent systems and individuals seeking to improve their efficiency, and decision-making abilities by providing an easily accessible source of information.

With the emergence of large language models (LLMs) such as GPT-3 [1] , a new use case for different kinds of knowledge bases has become apparent. Knowledge bases can be used to measure the understanding that a LLM has of different topics [2] . Knowledge bases can also be used to expand the functionality of LLMs [3] .

There are multiple large knowledge bases available. There are also multiple methods for creating new knowledge bases from new data and also methods for extracting new rules and information from existing knowledge bases.

The more well-known knowledge bases, such as ConceptNet [4] , Quasimodo [5] etc. contain a large amount of inference rules, which allow the inference of probable properties in the style of “Elephants have trunks” and “Airplanes have wings”.

These knowledge bases contain almost no rules which allow us to infer probabilistic classification, such as “If an animal has a trunk, that animal is likely an elephant” and “A machine that has wings is likely an airplane”. In other words: “If X is an animal, there is an N probability that X is a dog, M probability that X is a cat and a smaller probability that X is some other animal, unless it is known that it is some other animal”. The creation of such rules is the main goal of this thesis.

Despite the existence of multiple large knowledge bases and knowledge extraction methods, the amount of types of information in knowledge bases is extremely limited. Though these knowledge bases contain a large amount of inference rules, they lack probabilistic classification rules, which could be used alongside the more common rules for commonsense reasoning.

The probabilistic classification rules can be used for improving the performance of commonsense reasoning programs by providing rules which can be used to compare the relative probabilities of different events and statements. This can improve the reasoning programs ability of coming to plausible conclusions.

The goal of this thesis is to create a method for generating probabilistic classification rules from existing knowledge bases. The generated rules should be in the form "If X has wings, there is an N probability that X is a plane". By "probability" we actually mean probability relative to related rules. As a result of this thesis, a method for generating such rules will be established and a knowledge base of at least 5000 of these rules will be created and a subset of the rules will be evaluated.

The approach used in this thesis is based on statistical analysis. By utilizing the data already present in the knowledge bases, the probabilistic classification rules as well as initial supposability values for these rules will be generated. The supposability values, prepended to the probabilistic classification rules are used for measuring relative probabilities between similar rules.

External sources of data related to the rules such as word counts from Wikipedia will be used for improving the supposability values generated from the initial knowledge base. The resulting probabilistic classification knowledge base along with the data and code used to generate it are available in appendix 3.

Chapter 2 of this thesis provides an overview of existing methods for generating knowledge bases. Methods that utilize data mining, machine learning and natural language processing are presented. This chapter also includes a description of well-known knowledge bases, ConceptNet, WordNet and Quasimodo. All of these knowledge bases provide some data that will be used for creating the knowledge base of probabilistic classification.

Chapter 3 describes the data from which the probabilistic classification rules are generated. It describes the method used for generating the rules as well as the methods that were used to calculate and later improve the supposability values for the probabilistic classification rules.

Chapter 4 provides an overview of the results of the method used in this thesis. Different formulas were used for calculating the supposability values for the probabilistic classification rules. The results of the best performing formulas are analized. The formula that achieved the best results for improving the supposability values is used to update the supposability values of the rules in the probabilistic classification knowledge base and those updated rules are manually validated. This chapter also contains a discussion about what was achieved during this thesis and what could be improved in future works.

2 Background

This thesis deals with rule generation from existing knowledge bases, specifically generating rules, which can be used to infer probabilistic classifications. Among other things, knowledge bases are a useful component of search engines [6] . Therefore, there exist multiple methods for creating new knowledge bases and for extracting new rules from existing knowledge bases.

2.1 Data mining

Data mining is the process of discovering previously unknown information from existing data sources. Data mining can utilize pattern matching, statistics or mathematical theories [7] .

Data mining is used in a wide range of fields, including business, healthcare, finance, and science. It is particularly useful in situations where there is a large amount of data that needs to be analysed and understood, and can provide valuable insights and opportunities for improvement [7] .

Data mining involves using algorithms and statistical techniques to identify patterns and relationships within large sets of data and using the knowledge about the patterns and relationships to extract previously unknown information from data sources [7] . Data mining can be used to extract new information or rules from existing knowledge bases and other data sources by identifying hidden patterns and relationships that may not be apparent otherwise.

One example of creating a new knowledge base by using a data mining approach is described in [8] . The authors of this work explain that they set out to create a knowledge base of factual mistakes, an Anti-Knowledge base.

The authors describe that in order to gather data relevant to detecting factual mistakes they decided to mine for relevant information from Wikipedia updates, in the form of

pairs of sentences. They explain that each sentence pair contains a pre-update sentence and a post-update sentence. In particular interest to them are Wikipedia updates, which correct factual mistakes.

They explain that they chose Wikipedia as their source of data due the large variety of topics that it contains, its overall size and its active userbase. According to the authors, a large and active userbase means that any factual mistakes in Wikipedia are more likely to be spotted and corrected by the users, which in turn provides the authors with the data that they can use. They describe that detecting which updates contain factual corrections is one of the main difficulties of their task.

The authors explain that they focus on scanning for Wikipedia updates which contain claims that link an entity to a property in some way, as such claims are common. They explain that these pre-update and post-update sentence pairs were converted into subject, predicate, object triple pairs and that these triples would describe facts and pairs of these triples describe factual corrections. They explain that given two triples, one containing information about a topic before the information in the triple was updated and the other the other containing updated information about the same topic, it is possible, that the triple containing pre-update information contains a factual mistake.

The authors describe that in order to filer out Wikipedia updates containing factual corrections from the rest of the updates they used a combination of simple heuristics and a probabilistic machine learning model. They describe that one of the methods they use in order to detect which updates were meant to fix factual mistakes is comparing the pre-update and post-update triples. They compare each pair of triples and use an algorithm to detect swapped entities and numbers. The authors explain that checking for changed entities and numbers is a good indication of a factual mistake being corrected, as opposed to entirely new information being added to an article.

The authors describe using a distance measure heuristic for measuring the difference between the pre-update and post-update triples. Using this measure, the updates which fixed spelling mistakes, and thus have a small distance measure according to their heuristic could be ignored and not mixed in with updates that swapped entities or numbers, the authors wrote. They describe discarding updates, which fall below a threshold according to the distance measure.

Updates which replace words with synonyms were also detected using WordNet [9] and discarded, the authors write. After this, the authors devised a way to detect if an update that passed the previously described filtering methods was truly meant for correcting a factual mistake. To check this they use a version of the BERT [10] LLM to detect if the updates pre-update and post-update sentences contradict each other, they explain that if the sentences contradict each other then it is likely that the update was meant to correct a factual error.

From the remaining updates the authors discard the triples which are frequently updated and reverted. They explain that it is difficult to ascertain the truth from controversial topics which are frequently updated and in order to not generate any incorrect data, such contested updates are discarded.

The authors describe using a probabilistic model to determine the probability that the updates that made It past the previously listed filters are meant for correcting factual mistakes. They explain that once an update passes this model with a high enough probability then the triple of the pre-update sentence from this update will be added to the Anti-Knowledge base.

According to their explanation, they employ an unsupervised approach in their model due to the extensive size of the input data. Additionally, they clarify that relying solely on Wikipedia updates as input for the model is inadequate.

As per the authors' explanation, they utilize a snapshot of the entire Web text as a source of input for their probabilistic model. They explain that they mine the entire Web to count the occurrences of pre-update and post-update sentences. They use these sentence occurrence counts as input for their probabilistic model. The authors explain that the sentence counts are useful for the model because of a commonsense principle. They state, that sentences which contain factually correct information are more likely to appear than sentences that are factually incorrect.

The authors explain that they chose to search the Web for full sentences instead of triples because counting triples might lead to false positive matches for certain facts. Converting sentences about individuals with identical names into triples, for instance,

could result in the loss of crucial context, with information about one person being incorrectly attributed to another person with the same name, they add.

Finally the authors explain that using the method described above, they managed to mine more than 100000 factual mistakes from 16 terabytes of data from Wikipedia over a span of 24 hours. They converted these factual mistakes into subject, predicate and object triples and they used these triples to populate the Anti-Knowledge base.

2.2 Machine learning

Machine learning is a field of study that focuses on enabling computer systems to automatically learn and improve from experience without being explicitly programmed. It involves the development of algorithms and statistical models that enable a computer to identify patterns in data and make predictions or decisions based on those patterns. Machine learning algorithms can be categorized into three main types: supervised learning, unsupervised learning, and reinforcement learning [11] .

In supervised learning, the computer is trained on a labelled dataset, and it learns to make predictions based on that data. In unsupervised learning, the computer is given an unlabelled dataset, and it must find patterns and structure in the data on its own. In reinforcement learning, the computer learns to make decisions by interacting with an environment and receiving feedback in the form of rewards or penalties. Machine learning has many practical applications, including computer vision, natural language processing and speech recognition [11] .

Machine learning can also be used for generating rules, that can be used in knowledge bases. One such approach to generating rules is described in [12] . The authors of this work point out that there are numerous Artificial Intelligence (AI) systems available today thanks to the success of machine learning. The authors add that the effectiveness of these AI systems is limited because they are unable to provide explanations for their decisions. The authors then proceed to clarify that by learning relational rules that describe the information contained within the data, rule-based machine learning algorithms, unlike traditional models have the ability to generate rules that humans can better understand.

In order to solve the aforementioned problem, the authors of this work introduce a rule-based machine learning technique, FOLD-SE, which can produce a concise yet accurate set of rules from the input dataset. The authors explain that FOLD-SE is a rule learning algorithm that follows a top-down approach. This means that the algorithm starts by creating a potential rule and then proceeds to add literals from the featureset to expand the rule [12] .

According to the authors, utilizing a top-down approach in rule learning process provides advantages over a bottom-up approach. They explain that this advantage stems from the fact that a bottom-up approach does not perform as well on large datasets.

According to the authors, the majority of top-down rule-based machine learning algorithms, including FOLD-SE, rely on heuristics to direct the expansion of the rules' bodies. They add that variations of information gain heuristic are commonly used.

The authors explain that the choice of heuristic used for literal selection in rule-based machine learning algorithms has an effect on the degree of explainability of the resulting model. They explain that this is due to the fact that the chosen heuristic can influence the quantity and complexity of the generated rules.

The heuristic that the authors devised for use in FOLD-SE is Magic Gini Impurity (MGI), which is based on the Gini Impurity [13] heuristic. The authors note that by utilizing the MGI heuristic, they can reduce the number and complexity of the rules generated by FOLD-SE. They add that by using the MGI heuristic, the FOLD-SE algorithms performance remains comparable to the performance achieved by using the commonly used Information Gain heuristic.

FOLD-SE is capable of handling both categorical and numerical data values during the learning process, the authors write. They further add that FOLD-SE uses a comparison strategy that is designed for comparing categorical and numerical values. According to the authors, this allows FOLD-SE to handle values of different types and effectively learn from datasets that have features with both categorical and numerical values.

The authors describe using a parameter within the FOLD-SE algorithm to limit the minimum number of training examples that a rule can cover. They reason that this helps

to further reduce the complexity amount of rules generated by reducing overfitting of outliers.

Using the method described, the resulting rules could be modified as desired to fit a knowledge base.

2.3 Natural language processing

Natural Language Processing (NLP) is a subfield of artificial intelligence and computer science that deals with the interactions between computers and human languages. It involves using algorithms to interpret human language. NLP is used in a wide range of applications, such as machine translation, question answering, speech recognition, and information retrieval, among others [14] .

Natural language processing algorithms can also be utilised to create knowledge bases. One method for creating a knowledge base by using a natural language processing algorithm is described in [15] . The authors of this work explain that first it is important to gather the text-based natural language data that will be used to generate the knowledge base. They describe that an important step in this process is preprocessing the data.

The authors explain that correctly defining the beginnings and ends of all of the sentences in the gathered natural language data is a crucial step, as mistakes here could lead to severe errors during the following steps. They continue by stating that simply finding punctuation marks is insufficient for finding the endings of the sentences, as abbreviations and other mid-sentence texts ending with a punctuation mark would negatively effect the result of this method. The authors employ a pre-existing rule-based method that relies on a dictionary of abbreviations to accurately identify the sentence boundaries.

According to the authors the next step is dependency parsing. They explain that after the required text data has been gathered the next step is to choose a base dependency extraction algorithm and do any necessary preprocessing steps that are required by the base dependency extraction algorithm. They then describe three different types of dependency extraction algorithms. The three algorithms that they describe are based on

phrase structures, dependency grammars and machine learning. The authors of this work decided to use the StanfordNLP parser, because it has performed well on a related competition.

The next step according to the authors is constructing the graph structure, which is what makes up the knowledge base. They explain that this step is focused on extracting semantic relationships from the texts, using the parser selected in the previous step. The result of this step is the data that would later populate the new knowledge base, the structure of the result depends on the chosen parser.

Finally the authors state that the last step in the process of generating a knowledge base by using natural language processing is loading the generated data into a database using a query language, with this step a new knowledge base will have been created.

2.4 ConceptNet

ConceptNet is a knowledge graph, which uses labelled edges to connect words and phrases of the natural language [4] . ConceptNet was originally a representation of common sense rules, containing only crowd-sourced data, but now contains data gathered from multiple sources like crowd-sourcing, resources created by experts, etc. [4] . A small portion of the data from ConceptNet will be used for creating the knowledge base of probabilistic classification. Further information about this subset of rules is provided in chapter 3.1.

The aim of ConceptNet is to model common sense knowledge that facilitates language comprehension and enhances the performance of natural language processing applications. ConceptNet contains knowledge and relationships between words in multiple languages [4] .

The assertions in ConceptNet are treated as triples in the form (*dog, HasA, tail*). This triple represents the assertion “A dog has a tail.” [4] . The rules within the probabilistic classification knowledge base will utilize a similar structure.

ConceptNet uses a set of relations like *HasA*, *CapableOf*, etc. to represent the relationships between terms. The relations within ConceptNet are purposely similar to

the relations in WordNet. Some of the relations in ConceptNet, such as *RelatedTo* are symmetric, others like *CreatedBy* are directed [4] .

A symmetric relation is one which can be turned around, like *(parent, RelatedTo, child*). This triple represents the assertion "A parent is related to their child". Turning the triple around yields the triple *(child, RelatedTo, parent*), this represents the assertion "A child is related to their parent".

A directed relation means that head and tail parts of a rule can not be swapped around. This means that the rule (*dog, HasA, tail)* is true, but the rule (tail*, HasA, dog*) is not true.

The terms within ConceptNet are in a standardized form. The standardized terms are in all lowercase and all of the whitespaces have been replaced with underscores. ConceptNet also contains both terms that have been lemmatized and terms that have not been lemmatized [4] .

ConceptNet also imports knowledge from other knowledge bases and systems such as WordNet. The Nodes within ConceptNet are connected to terms within other sources via the *ExternalURL* relation. This relation holds a URL, which represents the term within the external knowledge source [4] .

2.5 Quasimodo

Quasimodo is a knowledge base of commonsense object properties. Quasimodo is also a methodology for extracting commonsense properties from the Web. The Quasimodo methodology is meant for extracting properties from question and answer forums and search engine query logs [5] .

The Quasimodo method gathers data in the form of questions from sites like Reddit and Quora, because questions can also convey knowledge. In order to gather more relevant data, the Quasimodo method utilizes data from the auto-completion suggestions of search engines. The questions gathered from the question and answer sites are transformed into statements and the statements are converted into *subject, predicate,*

object triples using an external tool. Various methods are then used to normalize the triples and some of the triples are removed entirely [5] .

In order to filter out unwanted data from the information gathering process, the gathered triples are inspected and scored using information from additional sources. In this process the triples of overly specific and overly generic assertions are filtered out. The triples' *predicate* and *object* co-occurrence data from Wikipedia, among data from different sources is used in this corroboration step [5] .

The co-occurrence of a pair of words generally refers to the number of times that the two words appear within the same context. The context in which the words are considered to be co-occurring can be defined based on the applications needs. For example, two words can be considered to be within the same context if they are in the same sentence, or if they are up to N words apart from each other in any given text. The heuristic can also be more complex.

The scores are then ranked. Using various formulas, the triples are given internal plausibility scores, and two conditional probability scores. These scores are used for ranking the triples [5] .

In order to further filter out unnecessary assertions, similar assertions are grouped together. The assertions which reflect the same commonsense property are grouped together using soft co-clustering [5] .

The Quasimodo knowledge base was created using this method. Using 7.5 million candidate triples extracted from various question and answer forums, which was filtered down to 2.3 million triples. The scores generated during the ranking process were used to evaluate the result, and the triples with the highest ranks were grouped together by similarity. The result is a knowledge base containing roughly 2.3 million assertion rules [5] .

A small subset of rules from the Quasimodo knowledge base will be used for the creation of the probabilistic classification knowledge base. Further detail on this subset of rules is provided in chapter 3.1.

2.6 WordNet

WordNet is a large database of English nouns, verbs, adjectives and adverbs [9] . These words are grouped into sets of synonyms [9] . The synonym sets also contain definitions for the words and short example sentences, utilizing the words in the synonym sets [9] .

The data and structure within WordNet make it a useful tool for natural language processing tasks [9] . These synonym sets are also used in this thesis for preprocessing the words extracted from the rules in ConceptNet and Quasimodo, more information about this topic can be found in chapter 3.2.

WordNet consists of four parts, one for each of nouns, verbs, adverbs and adjectives. The Words within each of these are grouped together based on their meanings, and the semantic relations of words are labelled [9] .

3 Methodology

This chapter describes the data and methods used for creating the knowledge base of probabilistic classifications. Since ConceptNet, Quasimodo etc. are large knowledge bases and contain information about many topics, in order to reduce development time, a smaller subset of these knowledge bases will be used to create the new knowledge base of probabilistic classifications.

3.1 Initial knowledge base

The rules used for generating the probabilistic classification knowledge base are taken from subsets of ConceptNet and Quasimodo. These subsets are cnet_05k.js and quasi_50k.js respectively. Both of these files contain 50000 rules concerning common concepts, which have been selected based on the commonness of the concepts involved and have been modified for use in commonsense reasoning by the supervisor of this thesis. These files can be accessed on the projects Github page, available in appendix 3.

The cnet_50k.js and quasi_50k.js knowledge bases are JSON files. These files are meant to be parsed by a custom parser, as they contains comments, starting with a double forward slash, which a regular JSON file should not contain. Figure 1 contains a small out-take of the comments found in cnet_50k.js, the comments in quasi_50k.js are also similar.

The rules within cnet_50k.js and quasi_50k.js are not ideal for creating a knowledge base of probabilistic classification as they are more difficult to parse and do not contain any more useful information that could not be extracted from the comments instead. The comments within these knowledge bases contain the original, unmodified rules from ConceptNet and Quasimodo, which can be used for generating a knowledge base of probabilistic classifications. These comments precede all of the modified rules in the knowledge bases.

In order to extract only the necessary data from cnet_50k.js and quasi_50k.js, a python script was used. This script parses the files and prints out from the files, only the lines which start with a double forward slash followed by a single whitespace. When printing the line, the double forward slash and whitespace preceding the body of the comment is stripped from the string in order to improve readability and simplify further parsing of the data. For that same purpose the |*True*, as can be seen on Figure 1, at the tail of the comments are also removed.

```
// people|CapableOf|think|True
// time|HasProperty|now|True
// people|HasProperty|good|True
```

Figure 1. Comments in cnet_50k.js.

Only the comments are extracted because they contain simple rules in the form *object*|*property*|*value* which can be easily read and modified later. The formatting of the extracted *object*, *property*, *value* triples can also be seen in Figure 2.

From the printing process, all of the lines containing the words *class element* are excluded, as the cnet_50k.js file contains close to 6000 instances of a comment containing only these two words. As this data is not useful for generating the new knowledge base, it will not be extracted.

The script was written to be able to handle multiple potentially blacklisted strings however the cnet_50k.js knowledge base only contains one type of comment which is not helpful for this thesis, the ones containing *class element*. quasi_50k.js does not contain any unnecessary comments. The rules gathered by this script are saved to separate text files, rules.txt and rules2.txt by using the pipe command. These files contain the triples gathered from cnet_05k.js and quasi_50k.js respectively.

The resulting knowledge bases, stored text files contain only *object*, *property*, *value* triples, as can be seen in Figure 2. This data is easy to process and provides a good starting point in the task of creating a knowledge base of probabilistic classifications. There are 98707 triples in total within the extracted data.

```
people|CapableOf|think
time|HasProperty|now
people|HasProperty|good
where|CapableOf|he_be
people|HasProperty|kind
```

Figure 2. Object, property, value triples extracted from cnet_50k.js.

3.2 Probabilistic classification rules

Now that the rules have been extracted from the comments within the cnet_50k.js and quasi_50k.js knowledge bases, the extracted data can be used to generate classification rules and then initial supposability values for each of the generated classification rules. The classification rules can be generated by flipping the extracted rules.

In order to flip the rules, a program which iterates over a list of triples and modifies each triple was created. For each *object* | *property* | *value* triple, this program creates a new rule in the form *X* | *property* | *value* => *X* | *isA* | *object.* Using this program to iterate over the rules from rules.txt and rules2.txt, the classification rules can be generated.

Now that the classification rules have been created, it is necessary to calculate supposability values for each of the classification rules. The supposability values can be thought of as relational probabilities, which are similar to probabilities, but not indicating actual probability. They can be used to compare the likelihood of similar rules applying to a given situation. Statistics and co-occurrence counts can be used to generate the initial supposability values.

The approach used in this thesis is inspired by the corroboration step of the Quasimodo method. In the Quasimodo method, co-occurrence data from Wikipedia along with a few other inputs were used to generate plausibility scores for the triples [5] .

In order to calculate supposability values for the classification rules, word counts and co-occurrences of words within the classification rules were used. To achieve this, a program which takes as input a list of classification rules was created. This program counts the number of times that each *object* appears in total within the set of rules. It also counts the co-occurrences of each *object, value* pair within the rules. This program

considers an *object* and *value* to be in a pair and thus co-occurring, if they are both present within the same rule. Each rule has exactly one *object* and one *value.*

The program then calculates supposability values for each of the classification rules. It calculates supposability values for each rule by dividing the *object*, *value* co-occurrence count by the *object* occurrence count given the *object, value* pair of a rule. These values are then prepended to the classification rules.

To test its effectiveness, this program was used to generate probabilistic classification rules from the rules.txt knowledge base, which contains the rules from cnet_50k.js. Figure 3 displays some of the resulting probabilistic classification rules.

```
0.125: X|CapableOf|think => X|isA|people
0.5: X|HasProperty|now => X|isA|time
0.017543859649122806: X|HasProperty|good => X|isA|people
1.0: X|CapableOf|he_be => X|isA|where
0.3333333333333333: X|HasProperty|kind => X|isA|people
```

Figure 3. Probabilistic classification rules generated from the rules.txt knowledge base.

In Figure 3 there is a rule *1.0: X|CapableOf|he_be => X|isA|where.* This rule, which is difficult to understand comes from the rules extracted from cnet_50k.js, which contains a few unintuitive rules from Conceptnet.

While at first glance this result may seem adequate as a starting point, there are some issues present within the new probabilistic classification knowledge base that should be addressed. To get a better understanding of the problem, rules containing similar statements are analysed.

By extracting and analysing the rules containing the *object* “car”, an issue with the supposability calculation becomes apparent. As can be seen from the supposability values and rules in Figure 4, it is equally likely that “A thing that has two wheels is a car.” and that “A thing that has four wheels is a car.”. Common sense indicates that a car should be more likely to have four wheels that two.

Figure 4 also displays some rules which only differ from each other by the forms of the words present in the rules. For example there is a rule stating that “A thing that has wheels is a car.”, and another rule stating that “A thing that has wheels is a cars.”. Both of these rules have a supposability value of 0.125.

In all cases where there exists an object in its singular and plural form, the co-occurrence counting will become less accurate because plural and singular words contribute to separate word counts. Counting plurals and singulars of the same word separately like this skews the results of the supposability calculation. Counting *car* and *cars* as the same object would increase the supposability, that "A thing that has wheels is a car.".

```
0.14285714285714285: X|HasA|four_wheels => X|isA|car
1.0: X|CapableOf|seat_riders => X|isA|car
0.125: X|HasA|wheels => X|isA|cars
0.125: X|HasA|wheels => X|isA|car
1.0: X|CapableOf|cost_more_than_houses => X|isA|cars
1.0: X|HasA|four_doors => X|isA|cars
0.3333333333333333: X|HasProperty|new => X|isA|car
0.005747126436781609: X|AtLocation|city => X|isA|car
0.14285714285714285: X|HasA|two_wheels => X|isA|cars
```

Figure 4. Probabilistic classification rules generated from the rules.txt knowledge base containing singular and plural versions of the same *object*.

In order to solve this problem, similar words need to be consolidated so that they can contribute to the same word counts during the word counting process. In order to achieve this, a function that can convert any given word into a more common variant of itself is required.

It is possible to lemmatize words in order to improve the results of counting co-occurrences [16] . Python's Natural Language Toolkit (NLTK) [17] provides a lemmatizing function which could be used for this purpose. The NLTK lemmatizing method uses data from WordNet to return the lemma of the input word, if the method fails to find the correct lemma from WordNet, it returns the original word itself [17] .

NLTK also provides multiple methods for stemming words [18] . Stemming could also be used for consolidating related words instead of lemmatizing them. There are three different stemming methods which are of interest to us. The three methods are *PorterStemmer*, *SnowballStemmer* and *LancasterStemmer*.

In order to find the most suitable method to consolidate the data, the performance of the different methods was assessed on a small set of input words. The stemming functions occasionally create words that are not part of the English language, as can be seen in

Table 1. Each stemming method stems some words correctly and some words incorrectly. Unlike the lemmatizing method, which leaves some words in their original form, but never returns a nonsensical word. The lemmatizer does not singularize some words.

Table 1. NLTK lemmatizer and stemmers' effects on different words.

Words	PorterStemmer	Snowball-Stemmer	Lancaster-Stemmer	Lemmatizer
cars	car	car	car	car
running	run	run	run	run
generously	gener	generous	gen	generously
vehicles	vehicl	vehicl	vehicl	vehicle
people	peopl	peopl	peopl	people

Because the stemming methods occasionally create nonsensical words, using them to consolidate the words within our knowledge base would not be ideal. This leaves the lemmatizer as the best available option.

In order to further consolidate the words that the lemmatizer left in their plural form, Python's pattern module's *singularize* function can be used. This function, as the name implies is used for converting a plural word into its singular counterpart. This function can be used to singularize the words that the lemmatizer left in the plural form. With these methods, the *object* and *value* words in rules.txt and rules2.txt can be consolidated by similarity.

A Python script was created to utilize the lemmatizing and singularizing functions to consolidate similar *object* and *value* words within the rules.txt and rules2.txt knowledge bases. This script also combines the two knowledge bases into a single knowledge base, singulars.txt.

By generating flipped rules from the lemmatized data within singulars.txt, the supposability values show a noticeable improvement over the previous iteration. The knowledge base generated in this step is saved to a text file, flipped.txt. Some of the

newly generated rules and supposability values can be seen in Figure 5. As can be seen from Figure 5, according to this new knowledge base of probabilistic classification with improved supposability values, “A thing that has four wheels is a car.” is more probable, than “A thing that has two wheels is a car.”. This follows common sense.

```
0.2857142857142857: X|HasA|four_wheel => X|isA|car
0.2222222222222222: X|HasA|wheel => X|isA|car
0.1111111111111111: X|HasA|wheel => X|isA|operational_car
0.125: X|HasA|two_wheel => X|isA|car
```

Figure 5. Lemmatized probabilistic classification rules from the flipped.txt knowledge base.

3.3 Improving the supposability values

This chapter provides an overview of the data used for improving the supposability values. In chapter 3.3.1, the data introduced in this chapter is analysed in further detail. The method that is used to apply this data in order to improve the plausibility of the initial supposability values is described in chapter 3.3.2.

Now that a knowledge base of probabilistic classification has been generated, it is time to further improve the plausibility of the initially calculated supposability values. While the supposability values generated from the word counts and co-occurrence counts within the knowledge base itself appear to be more accurate than assigning a constant supposability value to all of the rules, there is still room for improvement.

Some of the data that could be used to improve the supposability values are the co-occurrence counts for each *object*, *value* pair of the rules. A good source of data for word co-occurrences is Wikipedia, as it is a large and publicly available source of textual data. Wikipedia was also used for a similar purpose in the Quasimodo method [5] . The co-occurrence data will be taken from a dataset, wikimatrixrelatedtop20000. This is a dataset containing up to 1000 of the most common co-occurrences and co-occurrence counts for 20000 words. The co-occurrence data extracted from this dataset will be referred to as *object, value* co-occurrence. This dataset has been created by the supervisor of this thesis. It can be downloaded from the link in appendix 2.

Another source of data that can be utilized for calculating new supposability values for the probabilistic classification rules is important_words.txt. This is a dataset built from wikistats, frequency stats as well as common child words. The important_words.txt dataset consists of a collection of significant words and phrases, derived from a combination of common vocabulary used by children, English word frequencies, and Wikipedia access statistics. From the important_words.txt dataset, the sum of the Wikipedia access statistics and word frequencies for both the *object* and *value* words of the rules will be used as additional data to calculate new supposability values for the rules. These sums will be split into two separate datasets referred to as *object* importance and *value* importance. This dataset has also been created by the supervisor of this thesis, and can be downloaded from the same link as wikimatrixrelatedtop20000 in appendix 2.

There are 19702 rules within the flipped.txt knowledge base, for which there exists relevant *object, value* co-occurrence data within the wikimatrixrelatedtop20000 dataset. From the important words dataset, *object* and *value* importance can be found for all 19702 of the rules, for which there exist appropriate *object*, *value* co-occurrences. This means that this data can be used to improve the supposability values of 19702 rules, which is more than enough for the purposes of this thesis.

There are now three sets of data that can be used to modify the initial supposability values of the probabilistic classification knowledge base. To better understand the available co-occurrence and word importance data, as well as the initial supposability values of the rules within the probabilistic classification knowledge base, principal component analysis (PCA) [19] is utilized.

3.3.1 Principal Component Analysis

PCA is a statistical technique used for analysing tables of inter-correlated data. It is a linear transformation method, that can be used to extract only the most important data from a data table, by leaving out the data that contributes little information [19] .

The first step in PCA is data standardisation [19] . To standardise the data sets, scikit-learn [20] can be used. The scikit-learn package contains a method for standard-scaling, which allows all of the gathered data to be quickly scaled into a -1 to 1 range.

With the standardised data, it is possible to calculate the covariance matrix. The covariance matrix is an N * N matrix, where N is the number of variables in the input dataset [21] . The covariance matrix can be used to understand the covariance between the variables of the dataset that the matrix was calculated from [21] . To calculate the covariance matrix, NumPy [22] can be used as it contains a function for calculating a covariance matrix from a dataset.

The covariance of the initial supposability, co-occurrence and word count dataset can be seen in table 2. The numbers on the main diagonal of the covariance matrix represent variance within variables. The other numbers in the matrix represent covariance between two different variables [23] .

A positive covariance indicates that two variables are positively correlated, meaning that if the value of one variable increases, the value of the other variable will increase as well. A negative covariance means that two variables are negatively correlated, so if the value of one variable increases, the value of the other variable will decrease [23] .

Table 2 contains the covariance matrix of the initial supposability values calculated in chapter 3.2 and the three data sets discussed in chapter 3.3. As can be seen from table 2, the initial supposability value is positively correlated with all other variables. Co-occurrence is negatively correlated with both word importance values. This information can be used for modifying the formula for calculating new, more plausible supposability values for the probabilistic classification rules.

Table 2. Covariance matrix of initial supposability value, co-occurrence and word importances.

Variables	Initial suppos-ability	*Object, value* Co-occurrence	*Object* import-ance	*Value* import-ance
Initial suppos-ability	1.00005699	0.06624858	0.06762229	0.10386052
Object, value Co-occurrence	0.06624858	1.00005699	-0.07262421	-0.07236126
Object import-ance	0.06762229	-0.07262421	1.00005699	0.01669447
Value import-ance	0.10386052	-0.07236126	0.01669447	1.00005699

The next step in PCA is calculating the principal components in the form of eigenvectors and eigenvalues from the covariance matrix [21] . The eigenvectors and eigenvalues can be calculated using NumPy.

To visualize how much information each of the principal components holds, the principal components can be plotted as seen in Figure 6. Before plotting the principal components, it is common to sort the eigenvectors by their eigenvalues [21] .

Figure 6 displays the variance of each of the four principal components. This data can be used to ascertain how much information each principal component holds. This data can be used to remove the principal components or variables, which provide little unique information. Removing variables that do not provide much information can simplify later development, as there would be less data to work with.

As can be seen from Figure 6, the four principal components all hold a similar amount of information about data distribution. The principal component with the least information about data distribution holds only 33% less unique information than the principal component with the most info about data distribution. As all of the principal components contribute a non-trivial amount of information, none of them should be discarded without further consideration. Thus all four datasets discussed in this chapter will be taken into consideration when creating a formula for improving the initial supposability values.

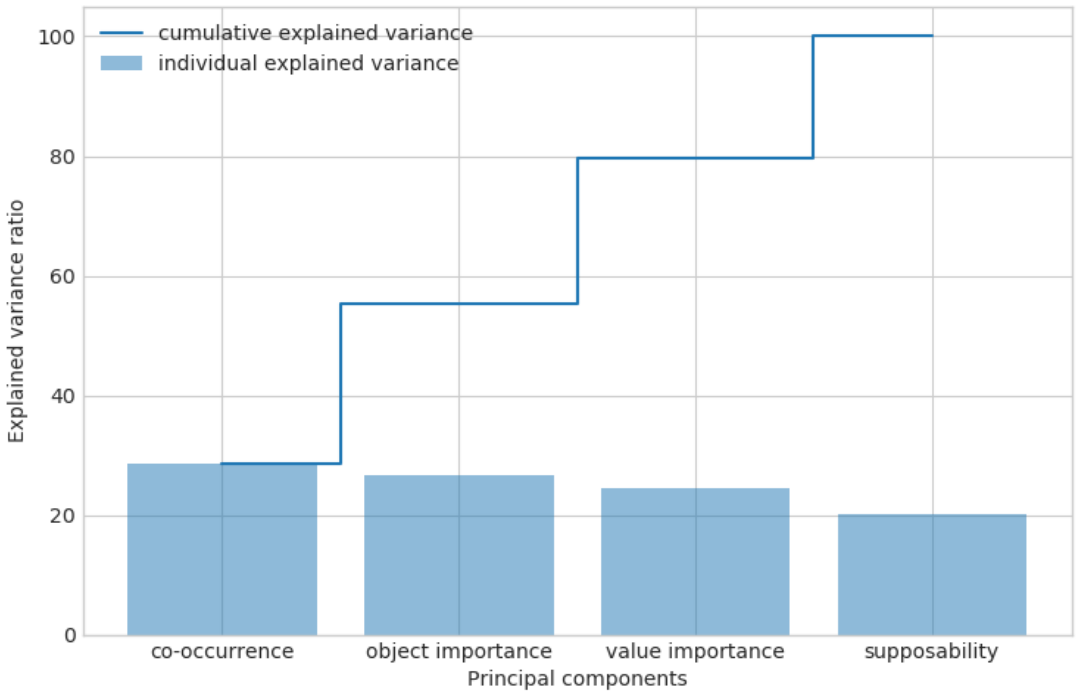

Figure 6. Principal components and how much information about data distribution they carry.

3.3.2 Applying the data

As all four sets of data discussed in the previous chapter contribute a considerable amount of information, none of them will be discarded at this time. In order to make the supposability values in the knowledge base of probabilistic classification more plausable, a suitable method must be devised for applying the available data to the existing supposability values in order to improve their plausibility. In order to improve the supposability values, different formulas which alter the initial supposability values of each rule using the previously gathered data are tested.

A Python script, calculate_supposability.py was created in order to be able to quickly test the effects of applying different formulas and data scaling methods to the data gathered in chapter 3.3. This script contains a function that takes two input variables and uses them to calculate new supposability values for the probabilistic classification knowledge base.

The first input that this function takes is a dataset containing the initial supposability values generated from flipped.txt, *object, value* co-occurrence counts from wikimatrixrelatedtop20000 and *object* and *value* importances from importantwords.txt. The second input variable that this function takes is another function, which describes the formula used to apply the data from the first input variable in order to calculate new supposability values. The function then uses the described formula and the data given as inputs to calculate new supposability values.

Using this method to modify the existing supposability values often results in some of the new values being either greater than 1 or lesser than 0. To make this data easier to parse and understand, it is scaled to a range between 0 and 1.

Now that a function has been created for quickly applying different formulas to calculate improved supposability values, a way to assess the quality of the results is needed. In order to achieve this, a validation dataset is required.

A dataset of word pair norms [24] can be used to validate the plausibility of the calculated supposability values. This dataset of word pair norms, double_words.txt, was created as part of an experiment where participants were asked to list multiple features or targets for each cue word presented to them [25] . The result of the experiment is a dataset that contains word pairs and the feature overlap of the rooted word pairs among other metrics [24] . Feature overlap is used to describe the quantity of features that two or more concepts have in common [24] . Feature overlap values typically range from 0 to 1. A feature overlap of 0 means that the two features are not correlated at all, and a feature overlap of 1 means that two features are perfectly correlated. An example of the data available in the double_words.txt dataset can be seen in Table 3.

Table 3 contains a cue word, in this case “abdomen”, the target words, such as “arms” and “bicep” and the feature overlaps of each cue word and target word as well as feature overlaps of the cue word root and target word root. A root word is a word that has been lemmatized.

As can be seen in Table 3, the words that make sense in a pair, such as abdomen and bowels have a high feature overlap, in this case, roughly 0.5. Words that are usually not associated with each other, such as abdomen and blink have a low feature overlap.

Table 3. Double_words.txt feature overlap for words and root words.

Cue	Target	Root	Raw
abdomen	arms	0.253597908	0.245690721
abdomen	bicep	0.232842922	0.117371443
abdomen	blink	0.018824004	0.012853065
abdomen	bowels	0.501481498	0.595626722
abdomen	burp	0.244023906	0.249929965

For validating the supposability values calculated by calculate_supposability.py, feature overlap between root words in the double_words.txt dataset will be used. The feature overlap of root words will be used, as the words in flipped_50k.txt have also been lemmatised and singularized. The supposability values calculated by calculate_supposability.py can be directly compared to the feature overlaps in double words, as they are both scaled between 0 and 1.

The word pairs in the double_words.txt dataset are matched together with rules from the probabilistic classification knowledge base, flipped.txt by comparing both words in a double_words.txt pair and the *object* and *value* words in the rule. There are roughly 1100 word pairs in the double words dataset that have matches in the probabilistic classification knowledge base. These 1100 word pairs and their feature overlaps can be used for validating the supposability values calculated by the different formulas.

In order to measure the similarity between the validation data and the calculated supposability values, a suitable heuristic is required. A distance measure that is commonly used for similarity measurements is cosine similarity [26] .

Cosine similarity is a distance measure that is commonly used as a heuristic in NLP tasks. It is a measure of similarity that calculates the cosine between two vectors [26] . When the vectors or datasets are similar, the cosine between them is 1, when the vectors are orthogonal to each other, the cosine is 0 [26] .

Cosine similarity, as a euclidean distance measure is not ideal for comparing probabilities. For comparing probabilities, improved sqrt-cosine (ISC) similarity is preferred [26] . Because the supposability values are a form of relational probability, the ISC similarity heuristic will be used to evaluate supposability values created by calculate_supposability.py. The implementation of the ISC similarity [26] can be seen in Figure 7. In Figure 7, the variable *confidences* contains the list of supposability values calculated by the designated formula. The variable *verification* contains the feature overlaps from double_words.txt

```
def i_sqrt_cos(confidences, verification):
    temp = 0
    for i in range(len(confidences)):
        temp += math.sqrt(confidences[i] * verification[i])
    return temp / (math.sqrt(sum(confidences)) *
math.sqrt(sum(verification)))
```

Figure 7. ISC calculation implentation in calculate_supposability.py.

Only 1100 of the 19702 probabilistic classification rules' supposability values could be compared to the validation data. This is because the rules which do not have any info about the feature overlap of their *object* and *value* within the double_words.txt dataset can not be compared directly against it.

The supposability values of the porbabilistic classification rules were matched to their respective word pair feature overlaps based on their *object* and *value*. The list of 1100 feature overlaps and the list of calculated supposability values are compared as vectors using the ISC metric.

Table 4 contains feature overlaps from the double_words.txt dataset and supposability values calculated through the application of two different formulas, formula A and formula B. Formula A does not do any changes to the supposability values, it can be represented with the following formula: A(x) = x. Formula B is a more complex formula, that modifies the supposability value by adding the co-occurrence value multiplied by the inverse of the divistion of the logarithm of the importance value of the *object* by the logarithm of the importance value of the *value.* This formula is explained in chapter 4 as formula (2). The data in this table reflects the values related to the two rules *X|CapableOf|study => X|isA|student* and *X|HasProperty|small => X|isA|short.*

The bottom row of Table 4 contains the ISC similarities, calculated by the code described in Figure 7, between the feature overlaps and supposability values. In the example given in this table, formula B performs much better than formula A. This is because the supposability values calculated by formula B achieved an ISC similarity of 0.99 to the validation data, whereas the supposability values calculated by formula A only reached an ISC similarity of 0.85. In order to calculate the most plausible supposability values for the probabilistic classification rules, the ISC similarity will be used to measure the results of different formulas. The formulas that are used to achieve the highest ISC similarities to the validation data will be considered for use in calculating the supposability values for the knowledge base of probabilistic classification.

Table 4. ISC similarities to validation data of supposability values calculated by two different formulas.

Object, value pair	Feature overlaps from double_words.txt	Supposability values calculated by formula A	Supposability values calculated by formula B
Student, study	0.496116453	0.42857142857142855	0.35145210786057113
Short, small	0.346729308	0.009523809523809525	0.39005906749121183
ISC similarity		0.853398653441032	0.9933791962934878

In order to further improve the supposability values calculated by calculate_supposability.py, the formula testing method applies weights to the variables in the dataset and modifies those weights recursively to optimize the similarity between the new supposability values and the feature overlap in double words, using the ISC similarity as a heuristic.

Figure 8 displays an example of the code used in calculate_supposability.py, specifically the function *adjust_supposability*, used for applying different formulas and data scaling techniques to the gathered data in order to calculate new supposability values. In this example the *data* variable is a list containing the different arrays of data, *data[0]* contains the initial supposability values from flipped.txt, *data[1]* contains the

object, value co-occurrences from wikimatrixrelatedtop20000, etc. Within The function *adjust_supposability,* the formula used for calculating new supposability values from the data is defined as a separate function. The *calc_rec* function applies the formula to the data and recursively changes the weights *c, co, o* and *v*, which are used as weights for the initial supposability values, *object, value* co-occurrences, *object* importances and *value* importances respectively. This function saves the highest ISC similarity to the validation data that it found to the global variable, *last* and the supposability values responsible for that similarity into *last_confs*.

```
def adjust_supposability(in_f, out_f, data):
    # The highest similarity with validation data and the
    # supposability for that similarity found by the recursive algorithm
    global last, last_confs

    # Standardize co-occurrences
    scaler = StandardScaler()
    data[1] = scaler.fit_transform(np.array([data[1], data[1]]).T).T[0]

    def apply_data(data, c, co, o, v, index):
        return (data[0][index] * c) + (data[1][index] * co)

    calc_rec(data, apply_data)
```

Figure 8. Function for applying different formulas and to the initial supposability value, co-occurrence and word importance data.

As a baseline, the similarity between the initial supposability values calculated in chapter 3.2 of this thesis, which have not been modified by any formula and the validation data from double_words.txt is 0.77 when using the ISC similarity measure.

Multiple formulas and data scaling methods were tested using the *adjust_supposability* function. Due to the small amount of validation data, a lot of the resulting supposability values appeared to be nonsensical, despite having a high similarity to the validation data. Often times most of the supposability values were near zero, despite the ISC similarity being around 0.85.

In order to improve the results further, the best performing formulas were saved and the weights applied to the data by those formulas were manually adjusted. As a result of the manual adjustments, some of the ISC similarities to the validation data became as high as 0.93 and the supposability values appeared to become more plausible as well.

In order to save the supposability values calculated by the *adjust_supposability* function, the function was modified to pair the new supposability values with their respective rules from flipped.txt. The rules along with their new supposability values are then written to a file, flipped_adjusted.txt. With this a knowledge base of 19702 probabilistic classification rules with adjusted supposability values is created.

Some of the better performing formulas are listed in chapter 4. In that chapter, the supposability values calculated by those formulas are also discussed.

4 Results

With the manual tweaking of of the weights and adjustments to the formulas, high ISC similarities to the validation data were achieved. As described in the previous chapter, the supposability values could still be nonsensical despite having a high ISC similarity to the validation data. In this chapter the best results of the supposability value calculation are discussed and the result with the most plausible supposability values is manually assessed.

Some of the formulas that were used to reached the highest ISC similarity to the validation data are:

- $S = supp + \log(importance_{value} \times 0.35) \times (cooc \times 0.05)$ (1)
- $S = supp + cooc \times \left(\frac{1}{\log(importance_{object}) * 0.5 \div \log(importance_{value})}\right)$ (2)
- $S = supp + cooc \times 0.55$ (3)

In the above formulas, *supp* represents the initial supposability value calculated for the probabilistic classification rule in chapter 3.2. In these formulas, *cooc* represents the *object, value* co-occurrence from wikimatrixrelatedtop20000. The *object* and *value* importance values from important_words.txt are represented by $importance_{object}$ and $importance_{value}$ respectively.

All of these formulas achieved ISC similarities close to 0.9 with the validation data. Before the supposability values from any of these formulas are manually checked, a cursory overview of the supposability values is acquired.

The supposability values calculated using formula (1) achieved an ISC similarity of 0.89 with the validation data. Close to 90% of the supposability values calculated using this formula were lower than 0.1. Because so many of the rules have such similar supposability values, it is difficult to compare them to each other. The skewed

distribution of supposability values, despite the high ISC score is likely the result of a small set of validation data. The rules that could be validated with the validation data were plausible and similar to the validation data, but the rest of the supposability values are nonsensical.

Using formula (2) to calculate the supposability values, an ISC similarity of 0.91 with the validation data was achieved. Most of the supposability values calculated using this formula are in the range 0.4 to 0.5. 34% of the supposability values belong to that range.

The supposability values calculated using formula (3) have an ISC similarity of 0.9 with the validation data. Similarly to the supposability values calculated by using formula (1), most of the supposability values calculated by this formula are smaller than 0.1.

4.1 Manual assessment

As both formula (1) and formula (3) produce results with heavily skewed results, the supposability values calculated by formula(2) are used for creating the final knowledge base of probabilistic classification. This knowledge base is saved to the flipped_adjusted.txt file. A subset of 200 rules from that knowledge base were evaluated. Additional rules from flipped_adjusted.txt were used for comparison with the evaluation rules where needed. The rules used for assessment can be found in the github page linked in appendix 3, under the file name assessment_data.txt.

Notice that the supposability values are not probabilities, they are relative values used for comparing and ordering similar rules. The supposability values are used for evaluating which rule is more likely to be true than another, related rule. For example, the rule *X|AtLocation|can => X|isA|pear* has a supposability value of 0.31 and the rule *X|AtLocation|can => X|isA|corn* has a supposability value of 0.39. This means that "An *X* that is in a can is more likely to be corn than a pear.". This does not mean that a thing that is inside of a can has a 39% probability of being *corn.* The supposability values should be used to compare the plausability of similar rules.

The rule *X|AtLocation|school => X|isA|child* states that "An entity that is at school is a child.". According to the rules within the knowledge base of probabilistic classification,

flipped_adjusted.txt, a child is the most likely entity to be at school, with a supposability of 0.29, followed closely by *study, room, friend, rule* and *principal*.

The rule with the smallest supposability for an entity that is located at school is *X*|*AtLocation*|*school => X*|*isA*|*bully.* This rule has a supposability of just 0.07. Other entities with a supposability of less than 0.1 for being at school are *playground, violin* and *printer.* These rules are also ordered in a plausible manner according to their supposability values.

It is also worth noting that in the flipped.txt knowledge base a *shark, sloth, mouse, gun* and *guinea pig* among other entities have a higher probability to be present at a *school* than a *bully*. In the flipped_adjusted.txt knowledge base, the nonsensical rules such as *X*|*AtLocation*|*school => X*|*isA*|*shark* have been removed, because there was no co-occurrence data available for the word pair *school, shark* within the wikimatrixrelatedtop20000 dataset.

The rules containing the statement *X*|*have*|*stripe* state, that "An *X* that has stripes is most likely a *bee*.", with a supposability of 0.4. According to these rules a *skunk* is least likely to have a *stripe*, with a supposability of 0.3. This can be seen in Figure 9.

There is an obvious flaw within this data. According to these rules a lion is more likely to have a stripes than a zebra. This goes against common sense, as zebras are generally known for their black and white stipes, whereas lions are not known for having stripes.

```
0.2921004518780414: X|have|stripe => X|isA|skunk
0.3027105795749384: X|have|stripe => X|isA|donkey
0.30582785009117736: X|have|stripe => X|isA|straw
0.3120176669079754: X|have|stripe => X|isA|cheetah
0.3280229376263086: X|have|stripe => X|isA|zebra
0.33970407950323617: X|have|stripe => X|isA|lion
0.35167689746972464: X|have|stripe => X|isA|tiger
0.405448141324551: X|have|stripe => X|isA|bee
```

Figure 9. Sorted rules containing the statement *X*|*have*|*stripe* in the flipped_adjusted.txt knowledge base.

In the flipped.txt knowledge base, all of the *X*|*have*|*stripe* rules have a supposability of 0.042. This means that the adjustments made to the supposability values by formula (2) have greatly improved the plausibility of the supposability values of the probabilistic

classification rules, however as discussed earlier some errors remain within the final knowledge base.

According to the rules containing the statement *X|AtLocation|can, X* is commonly a food item like *water, corn, meat, soda,* etc. *X* can also be *paint* or *worms*. This can be seen in Figure 10. The supposability values of these rules follow from the common sense of the author of this thesis, with water and food being the most likely items to be inside of a can.

```
0.35586000119048555: X|AtLocation|can => X|isA|pear
0.36519715486124865: X|AtLocation|can => X|isA|spam
0.36929648970886886: X|AtLocation|can => X|isA|worm
0.3711331495767831: X|AtLocation|can => X|isA|pea
0.3711781544155582: X|AtLocation|can => X|isA|mushroom
0.3765885779555174: X|AtLocation|can => X|isA|soda
0.3776773088954205: X|AtLocation|can => X|isA|tuna
0.3819477388232937: X|AtLocation|can => X|isA|lip
0.3849773794866281: X|AtLocation|can => X|isA|coke
0.38615585596022917: X|AtLocation|can => X|isA|soup
0.3890010057753332: X|AtLocation|can => X|isA|bean
0.38920909981694946: X|AtLocation|can => X|isA|garbage
0.3914741947819344: X|AtLocation|can => X|isA|liquid
0.39358343050318667: X|AtLocation|can => X|isA|vegetable
0.40743716710425937: X|AtLocation|can => X|isA|beer
0.4158451439185088: X|AtLocation|can => X|isA|coffee
0.44775755533108613: X|AtLocation|can => X|isA|paint
0.44816727867977646: X|AtLocation|can => X|isA|drink
0.4483344062077903: X|AtLocation|can => X|isA|meat
0.4510770933732648: X|AtLocation|can => X|isA|corn
0.4512574839232868: X|AtLocation|can => X|isA|food
0.45329432704425365: X|AtLocation|can => X|isA|water
```

Figure 10. Sorted rules containing the statement *X|AtLocation|can* in the flipped_adjusted.txt knowledge base.

X|HasA|parent => X|isA|child is a rule with a 0.3 supposability**.** It has a slightly higher supposability than the rule *X|HasA|parent => X|isA|person*, which has a supposability of 0.296. That rule in turn has a higher supposability than *X|HasA|parent => X|isA|everyone*, which has a supposability of 0.294. This order follows logically from the common sense of the author of this thesis.

According to the rules present in Figure 11, a thing that is *mean* is most likely a person. A *dog* is more likely to be mean than a *human* according to these rules. Following the

common sense of the author of this thesis, a *human* should be more likely to be *mean* than a dog. The supposability of the rule *X|HasProperty|mean => X|isA|human* is low likely because the word par *mean, human* has a low co-occurrence count within the wikimatrixrelatedtop dataset. This means that formula (2) will increase the initial supposability value of this rule only by a small amount compared to other rules.

```
0.3821224821163774: X|HasProperty|mean => X|isA|swan
0.4558573607647971: X|HasProperty|mean => X|isA|human
0.46726893322625296: X|HasProperty|mean => X|isA|dog
0.46726893322625296: X|HasProperty|mean => X|isA|dog
0.46761059156582585: X|HasProperty|mean => X|isA|person
```

Figure 11. Sorted rules containing the statement *X|HasProperty|mean* in the flipped_adjusted.txt knowledge base.

Figure 12 lists the rules containing the statement *X|AtLocation|family*. According to the authors common sense there is only one rule with an incorrect supposability value within this list. That erroneous rule is *X|AtLocation|family => X|isA|koala.* This rule should have a smaller supposability value than the rule *X|AtLocation|family => X|isA|intimacy.*

```
0.40480280885372716: X|AtLocation|family => X|isA|intimacy
0.41335395751618687: X|AtLocation|family => X|isA|koala
0.4274853513694161: X|AtLocation|family => X|isA|sibling
0.4671274218559714: X|AtLocation|family => X|isA|support
0.4754013974476241: X|AtLocation|family => X|isA|human
0.48134332938211194: X|AtLocation|family => X|isA|brother
0.4833116994261272: X|AtLocation|family => X|isA|love
0.486118968524759: X|AtLocation|family => X|isA|child
0.494478988495128: X|AtLocation|family => X|isA|person
```

Figure 12. Sorted rules containing the statement *X|AtLocation|family* in the flipped_adjusted.txt knowledge base.

Another example of nonsensical supposability values can be seen in Figure 13. As can be seen from Figure 13, the supposability value of *X|have|layer => X|isA|onion* is a lot lower than the supposability value of *X|have|layer => X|isA|earth*. Following the common sense of the author of this thesis, a thing that has a *layer* should more likely be an *onion* than the *Earth*.

```
0.27949402683454544: X|have|layer => X|isA|onion
0.2995453572947363: X|have|layer => X|isA|jupiter
0.3820014884610884: X|have|layer => X|isA|earth
```

Figure 13. Sorted rules containing the statement *X|have|layer* in the flipped_adjusted.txt knowledge base.

Within the knowledge base are also some rules which can not be compared to any other rule from the same knowledge base. Rules such as *X|be called|vector => X|isA|mosquito* can not be compared to other rules as there are no other rules containing the statement *X|be called|vector*. Rules that can not be compared against other, similar rules are less useful as they do not provide any information that could not be gathered from most other knowledge bases.

4.2 Discussion

As can be seen from the results discussed in the previous chapter, the probabilistic classification rules generated by the method developed for this thesis, contain a non-trivial amount of useful information. These rules can be used to infer probabilistic classifications such as “An X that has stripes is likely a bee, and that X is a little bit less likely a tiger.”, etc.

The assessment of the rules also revealed that some of the generated probabilistic classification rules contain some errors. Some of the generated rules have supposability values that rank them as less likely to occur than they should be, compared to related rules. This creates situations where according to this knowledge base, “An X that has layers is most likely the Earth and that X is least likely to be an onion.”. While it is factually correct that the Earth has layers, when thinking about things that have layers, an onion should usually be the thing that comes to mind before the Earth.

The final problem with the new probabilistic classification knowledge base is that some rules can not be compared against any other rules. There are some rules which contain statements which no other rule has, such as *X|be called|vector => X|isA|mosquito*. While this rule alone can be used to infer that “A thing that is called a vector is most likely a mosquito.”, because there are no possible comparisons, it is equally likely that “A mosquito is the least likely X to be a vector.”.

Alltogether there are 19702 probabilistic classification rules within the newly generated knowledge base, flipped_adjusted.txt. Of those rules, 16792 of them can be compared against at least one other rule. There are a total of 1973 unique statements, in the form *X*|<*property*>|<*value*> that are used by at least two different rules.

One of the improvements that could be made to the method developed in this thesis is using a machine learning algorithm to apply the co-occurrence and word importance data to the initial supposability values. The method developed in this thesis uses formulas created through trial and error in order to efficiently apply the available data to improve supposability values, which are used to order and compare similar rules.

One other improvement that could be made is using a larger dataset of co-occurrence counts. The wikimatrixrelatedtop20000 dataset contained co-occurrence data for only 20% of the rules extracted from cnet_50k.js and quasi_50k.js. A custom method could be developed to gather only the necessary co-occurrences from Wikipedia.

The probabilistic classification rules discussed in this thesis were generated from a knowledge base of only 100000 rules. A small knowledge base was used during development process in order to simplify development and debugging. With minor adjustments to the developed code, larger knowledge bases can be used as input instead.

5 Summary

There are multiple knowledge bases containing large amounts of inference rules. In order to provide new information, not usually available within commonly used knowledge bases, a method for generating probabilistic classification rules from existing knowledge bases was created.

The developed method modifies existing inference rules in the form of *object, property, value* triples in order to create classification rules in the form *X*|*property*|*value* => *X*|*isA*|*object*. The *object* and *value* occurrence and co-occurrence data gathered from the classification rules is then used to calculate supposability values for the classification rules. The supposability values are appended to the classification rules, creating probabilistic classification rules. Additional co-occurrence and word importance data from Wikipedia is then used to improve plausibility the supposability values.

A subset of 200 of the generated probabilistic classification rules was manually assessed. The results and possible improvements to the method were presented.

References

[1] L. Floridi and M. Chiriatti, “GPT-3: Its Nature, Scope, Limits, and Consequences,” Minds & Machines, vol. 30, no. 4, pp. 681–694, Dec. 2020, doi: 10.1007/s11023-020-09548-1.

[2] P. Sahu, M. Cogswell, Y. Gong, and A. Divakaran, “Unpacking Large Language Models with Conceptual Consistency.” arXiv, Sep. 29, 2022. doi: 10.48550/arXiv.2209.15093.

[3] Y. Sun, Y. Xu, C. Cheng, Y. Li, C. H. Lee, and A. Asadipour, “Explore the Future Earth with Wander 2.0: AI Chatbot Driven By Knowledge-base Story Generation and Text-to-image Model,” in *Extended Abstracts of the 2023 CHI Conference on Human Factors in Computing Systems*, in CHI EA ’23. New York, NY, USA: Association for Computing Machinery, Apr. 2023, pp. 1–5. doi: 10.1145/3544549.3583931.

[4] R. Speer, J. Chin, and C. Havasi, “Conceptnet 5.5: An open multilingual graph of general knowledge,” in *Proceedings of the AAAI conference on artificial intelligence*, 2017.

[5] J. Romero, S. Razniewski, K. Pal, J. Z. Pan, A. Sakhadeo, and G. Weikum, “Commonsense properties from query logs and question answering forums,” in *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, 2019, pp. 1411–1420.

[6] A. Bernstein, E. Kaufmann, and C. Kaiser, “Querying the semantic web with ginseng: A guided input natural language search engine,” in *15th Workshop on Information Technologies and Systems, Las Vegas, NV*, Citeseer, 2005, pp. 112–126.

[7] M.-S. Chen, J. Han, and P. S. Yu, “Data mining: an overview from a database perspective,” *IEEE Transactions on Knowledge and data Engineering*, vol. 8, no. 6, pp. 866–883, 1996.

[8] G. Karagiannis, I. Trummer, S. Jo, S. Khandelwal, X. Wang, and C. Yu, “Mining an" anti-knowledge base" from Wikipedia updates with applications to fact checking and beyond,” *Proceedings of the VLDB Endowment*, vol. 13, no. 4, pp. 561–573, 2019.

[9] G. A. Miller, “WordNet: An electronic lexical database”. MIT press, 1998.

[10] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.” arXiv, May 24, 2019. doi: 10.48550/arXiv.1810.04805.

[11] M. I. Jordan and T. M. Mitchell, “Machine learning: Trends, perspectives, and prospects,” *Science*, vol. 349, no. 6245, pp. 255–260, 2015.

[12] H. Wang and G. Gupta, “FOLD-SE: An Efficient Rule-based Machine Learning Algorithm with Scalable Explainability.” arXiv, Jan. 10, 2023. doi: 10.48550/arXiv.2208.07912.

[13] E. Laber, M. Molinaro, and F. M. Pereira, “Binary Partitions with Approximate Minimum Impurity,” in *Proceedings of the 35th International Conference on Machine*

Learning, PMLR, Jul. 2018, pp. 2854–2862. Accessed: May 09, 2023. [Online]. Available: https://proceedings.mlr.press/v80/laber18a.html

[14] E. D. Liddy, “Natural language processing,” 2001.

[15] A. A. Maksutov, V. I. Zamyatovskiy, V. N. Vyunnikov, and A. V. Kutuzov, “Knowledge Base Collecting Using Natural Language Processing Algorithms,” in *2020 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus)*, Jan. 2020, pp. 405–407. doi: 10.1109/EIConRus49466.2020.9039303.

[16] K. Schouten, O. van der Weijde, F. Frasincar, and R. Dekker, “Supervised and Unsupervised Aspect Category Detection for Sentiment Analysis with Co-occurrence Data,” *IEEE Transactions on Cybernetics*, vol. 48, no. 4, pp. 1263–1275, Apr. 2018, doi: 10.1109/TCYB.2017.2688801.

[17] N. Hardeniya, J. Perkins, D. Chopra, N. Joshi, and I. Mathur, “Natural Language Processing: Python and NLTK”. Packt Publishing Ltd, 2016.

[18] M. Augat and M. Ladlow, “CS65: An NLTK Package for Lexical-Chain Based Word Sense Disambiguation,” *Word J. Int. Linguist. Assoc*, 2004.

[19] H. Abdi and L. J. Williams, “Principal component analysis,” *Wiley interdisciplinary reviews: computational statistics*, vol. 2, no. 4, pp. 433–459, 2010.

[20] “scikit-learn,” scikit-learn. [Online]. Available: https://scikit-learn.org/stable/. [Accessed: 22-Apr-2023].

[21] “Principal Component Analysis in 3 Simple Steps,” Sebastian Raschka. [Online]. Available: https://sebastianraschka.com/Articles/2015_pca_in_3_steps.html#covariance-matrix. [Accessed: 03-Apr-2023].

[22] “NumPy,” NumPy. [Online]. Available: https://numpy.org. [Accessed: 24-Apr-2023].

[23] “A Step-by-Step Explanation of Principal Component Analysis (PCA),” builtin.com. [Online]. Available: https://builtin.com/data-science/step-step-explanation-principal-component-analysis. [Accessed: 13-Apr-2023].

[24] “Word Pair Norms,” *WordNorms.com*. [Online]. Available: https://doomlab.shinyapps.io/double_words/. [Accessed: 29-Mar-2023].

[25] E. M. Buchanan, K. D. Valentine, and N. P. Maxwell, “English semantic feature production norms: An extended database of 4436 concepts,” *Behavior Research Methods*, vol. 51, pp. 1849–1863, 2019.

[26] S. Sohangir and D. Wang, “Improved sqrt-cosine similarity measurement,” *Journal of Big Data*, vol. 4, no. 1, pp. 1–13, 2017.

Appendix 1 – Non-exclusive licence for reproduction and publication of a graduation thesis[1]

I Renet Rämman

1 Grant Tallinn University of Technology free licence (non-exclusive licence) for my thesis “Generating Probabilistic Classification Rules from Existing Knowledge Bases”, supervised by Tanel Tammet.

1.1 to be reproduced for the purposes of preservation and electronic publication of the graduation thesis, incl. to be entered in the digital collection of the library of Tallinn University of Technology until expiry of the term of copyright;

1.2 to be published via the web of Tallinn University of Technology, incl. to be entered in the digital collection of the library of Tallinn University of Technology until expiry of the term of copyright.

2 I am aware that the author also retains the rights specified in clause 1 of the non-exclusive licence.

3 I confirm that granting the non-exclusive licence does not infringe other persons' intellectual property rights, the rights arising from the Personal Data Protection Act or rights arising from other legislation.

07.05.2023

1 The non-exclusive licence is not valid during the validity of access restriction indicated in the student's application for restriction on access to the graduation thesis that has been signed by the school's dean, except in case of the university's right to reproduce the thesis for preservation purposes only. If a graduation thesis is based on the joint creative activity of two or more persons and the co-author(s) has/have not granted, by the set deadline, the student defending his/her graduation thesis consent to reproduce and publish the graduation thesis in compliance with clauses 1.1 and 1.2 of the non-exclusive licence, the non-exclusive license shall not be valid for the period.

Appendix 2 – Wikipedia data download

Direct download link:

http://dijkstra.cs.ttu.ee/~tammet/wikicooccurrence.tar.gz

Appendix 3 – Github repository with source code

Github link:

https://github.com/RenetRamman/Probabilistic-classification-knowledge-base