Radboud Universiteit

Query by Navigation over Legal Documents: an Automated Pipeline using Formal Concept Analysis

Edwin Wenink
s4156072

Bachelor Thesis in Artificial Intelligence

Radboud University

June 28, 2019

supervised by
Franc Grootjen Ph.D
Radboud University

Reading committee:
Franc Grootjen Ph.D
Ceci Verbaarschot MSc

Contents

1 Introduction

Despite the availability of highly structured databases for some applications, for most companies it nevertheless holds that the major portion of valuable information resides in collections of relatively unstructured text documents. One example is the publishing company Wolters Kluwer, which maintains a large corpus of about half a million legal documents that includes jurisprudence, Dutch law, several legal magazines, legal textbooks and legal commentaries. As a service to legal professionals, Wolters Kluwer offers a search engine to retrieve information from these documents by querying keywords.

Such keyword-based information retrieval systems assume that the content of documents can be meaningfully captured by a set of keywords. If a keyword is indeed an adequate indication of what a document is about, then any document containing the queried term is likely to be meaningful for the query (Grootjen, 2005, p. 6). Although these systems work well in practice, a keyword-based query will often return a large amount of documents that the end user has to further inspect manually. Even though there are numerous methods for ranking search results on their relevance for a given query, such as the well-known TF*IDF relevancy measure (Spark Jones, 1972), manually browsing through search results nevertheless takes a lot of time. In the case of legal professionals who work at a high hourly rate there is high pressure to quickly assess the quality of a particular result. But this need for efficiency is at the same time at odds with the need to not miss any crucial information. In other words, within the high information need of legal professionals we can further distinguish a demand to not miss crucial information (high *recall*), while simultaneously not wasting any time on less relevant documents (*accuracy* and *relevance*). The practical motivation behind this thesis project is to support users with a high information need, such as lawyers, in navigating through search results efficiently.

For that purpose, this thesis project investigates and implements a pipeline using Natural Language Processing (NLP) and Formal Concept Analysis (FCA), which is a combination that has attracted significant research interest (e.g. Cimiano et al., 2003, 2005; Ilieva and Ormandjieva, 2007). Formal Concept Analysis is an unsupervised machine learning technique that will be used to automatically extract (formal) concepts from legal texts of the Wolters Kluwer database, in order to build a conceptual taxonomy that represents semantic relations between documents. Because this conceptual structure is based on the *contents* of documents, the need to manually inspect documents is reduced. Instead, the proposed pipeline gives users a meaningful abstraction of the contents of search results with which they can further specify their search for relevant information. One could for example employ query by navigation through the semantic structure (Grootjen, 2005, p.58), give relevance feedback (Grootjen, 2005, chap.6), or make suggestions for query refinement to related sub-concepts. This thesis investigates query by navigation, in which users can themselves interactively browse the conceptual hierarchy efficiently based on what they themselves deem relevant. Moreover, this proposed pipeline is emancipated from human labour because it is automated.

Applying NLP and concept-extraction techniques such as FCA in the legal domain is an active line of research that this thesis hopes to contribute to (e.g. Wyner et al., 2010). When these techniques meet legal praxis, they can lead to new insights into "the interconnections among authoritative legal texts and the behavior of courts, legislatures, lawyers, and other legal professionals" (Conrad and Branting, 2018). Many design decisions have to be taken in the implementation of various steps of the proposed automated pipeline, and they will be explained and justified in what follows.

There are three main research questions that this thesis addresses:

1. How can Formal Concept Analysis be used meaningfully with a large corpus of documents (0.5

million)?

2. How do we choose relevant formal context attributes to adequately represent the content of a set of documents?
3. How do different choices for formal context attributes affect concept navigation by the user?

2 Data

All analyses of this thesis are performed on data provided by the publishing company Wolters Kluwer. Wolters Kluwer maintains a large dataset of legal documents, containing amongst other things Dutch jurrisprudence, Dutch legal text books, the Dutch law, some international law relevant for Dutch legal practice, and yearly editions of legal journals. Legal professionals can purchase a license to access and search these documents to support their legal practices. To be able to work with this data for my thesis, I signed a Non-Disclosure Agreement (NDA).

The documents in this database are stored as semi-structured xml documents, sometimes containing links to relevant pdf documents with photocopies of printed editions. Each document contains xml tags indicating a hierarchical categorization of that document in terms of a general rubric (e.g. civil law), a main discipline therein (e.g. inheritence law), subtopics (e.g. testaments), and then relevant keywords (e.g. "codicil"). These categories are associated with an ID in a large thesaurus. In total the database provided by Wolters Kluwer contains 496.311 $\approx$ 0.5 million xml documents (excluding pdf files).

3 Methods and background information

3.1 What are concepts and how will we use them?

There are many philosophical answers to the question what a concept *is*. Even though there are fundamental disputes about this *ontological* question, there is a general agreement that a concept is a *representation*, irregardless of whether this representation is for example mental or abstract. This thesis deals with concepts, but will do so in a quite specific manner. First of all, with respect to the ontology of concepts this thesis shall be agnostic: it will for example not claim that concepts exist as mental representations. Secondly, we thus also do not speak informally of a concept as just "an idea". Thirdly, instead we will define a concept formally and use it as a formal tool of analysis.

For the latter formalization we will investigate an unsupervised machine learning technique called Formal Concept Analysis (FCA), which has a mathematically well-defined definition of a concept, that nevertheless fits the basic philosophical intuitions about what the essential structure of a concept is, namely a relation between its extension (what objects to we indicate with the concept) and intension (their attributes). Formal Concept Analysis was coined by Rudolf Wille as an attempt to use mathematics to represent real-world meanings, rather than just performing "elaborate mental gymnastics" (Wille, 2009). As a formal tool, FCA can be applied to various domains and various problems. We will explore FCA as a tool for information retrieval from a corpus of texts.

Texts are full of concepts. In some cases a concept can be named with a single word. For example, when we encounter the word "cat" in a text, this is a so-called "lexical concept" that either refers to actual cats, mental representations thereof, or perhaps some "abstract" cat. However, simultaneously this example illustrates another aspect of concepts: they have super- and subrelations to other concepts. A cat is, as we all know, a suitable pet (a superconcept), but is also an animal (an even more general superconcept, because not all animals can be pets). However, a "Siamese" is a particular type of cat

(a subconcept). Concepts thus have a particular "scope" that can be more general or more specific. Interestingly, there seems to be an inverse relationship between how many objects are part of a concept (its extension), and the amount attributes that the concept expresses (its intension). On the most general level (large extension: all animals), relatively little attributes need to be shared between various "animal instances" for us to call them animals. However, when we speak of cats specifically (smaller extension), we specify way more attributes (bigger intension): not just that it is alive and can move autonomously, and that it feeds on organic matter (an animal, basically), but also that it meows, is small, likes to sleep, has whiskers and retractable claws, and that it is carnivorous etc.

A cat is a subset of the larger set of animals, but the properties of animals are a subset of the properties of cats. This gives us a clue about how we can navigate from subsets to supersets and vice versa. To go from a subset to a superset, we either add objects to the concept's extension, or we remove attributes from the concepts' intension. Vice versa for the other direction. That is, based on a conceptual hierarchy, we can *formalize a form of conceptual navigation.*

So far, we have discussed concepts simply as linking some entity (e.g. a cat) to a set of attributes. However, companies such as Wolters Kluwer deal with large text corpora, and a main concern for them is how to retrieve relevant documents for clients and allow those clients to navigate through those documents. We will address this concern by trying to build a conceptual structure of a *corpus of documents* rather than on objects occurring in a single document, which allows navigation through those documents based on this conceptual structure. This poses two distinct challenges:

1. What is a "concept" in this type of information retrieval problem?
2. How do we manage the increased complexity due to the size of those corpa?

In the following, I will first give a brief explanation of Formal Concept Analysis and an indication of its relevance. In this section it will also become clear how we can define a conceptual hierarchy on a larger set of documents, as well as how navigation through this conceptual structure can aid the user in finding relevant documents. Secondly, I will address the added complexity that comes with a large corpus size, and how this limits the effectiveness of applying FCA in these scenarios. I explore potential countermeasures to keep FCA feasible and justify the approach taken in this thesis.

3.2 Short explanation of Formal Concept Analysis

Ignatov describes FCA as such:

> FCA is concerned with the formalisation of concepts and conceptual thinking and has been applied in many disciplines such as software engineering, machine learning, knowledge discovery and ontology construction during the last 20-25 years. Informally, FCA studies how objects can be hierarchically grouped together with their common attributes.“ (Ignatov, 2015, p. 1)

The formalisation of concepts that FCA offers is based on mathematical lattice theory. The goal here is not to exhaustively summarize all mathematical background, nor provide proofs. However, there are some important definitions without which it is not possible to understand FCAs capabilities and limitations. I specifically focus on the definitions that are required for understanding what a formal concept is. I base myself on the extensive FCA tutorial of Ignatov (Ignatov, 2015, p. 3-19).

First of all, in order to find concepts we first need to define a *formal context*, which contains the ingredients we already shortly indicated to be relevant for a formal concept: a set of objects whose subsets are possible extensions, and a set of attributes defining the possible intensions. We will define a formal concept explicitly below, but we already know that at least we also must know the relations between objects and attributes. A formal context is thus defined as:

$$C = (O, A, R)$$

Where O is a set of *objects*, A is a set of *attributes* and R is a set of *relations* defined on pairs (o,a) $\in R$.

Given these relations between objects and attributes, we can define two *derivation operators*, namely one that returns for a set of objects the set of attributes that they share, and another one that returns for a set of attributes the set of objects that they have in common. This gives us the following formal definitions.

For some subset of objects, $X \subseteq O$, we can derive a set X' such that

$$X' := \{a \in A | (o, a) \in R \; for \; all \; o \in X\}$$

and likewise for some subset of attributes, $Y \subseteq A$, we can derive Y' such that:

$$Y' := \{o \in O | (o, a) \in R \; for \; all \; a \in Y\}$$

These operators are *concept-forming* because they are the basis of the formal definition of a concept. A *formal concept* within a particular formal context is a pair (X, Y) where $X \subseteq O$ and $Y \subseteq A$ such that $X' = Y$ and $Y' = X$.

That is, a formal concept consists of a set of objects (its extent) and a set of attributes (its intent) in such a way that the extent is the derivation of the intent, and the intent is the derivation of the extent. As such, it is conform an old philosophical tradition as well as an international standard of the terminology of concept:

> This definition says that every formal concept has two parts, namely, its extent and intent. This follows an old tradition in the Logic of Port Royal (1662), and is in line with the International Standard ISO 704 that formulates the following definition: A concept is considered to be a unit of thought constituted of two parts: its extent and its intent (Ignatov, 2015, p. 7).

It is important that for any subset $X \subseteq O$ we can potentially find a concept, which gives a large amount of potential concepts as the set of objects O becomes larger (and likewise for the set of attributes A). The set of all concepts of a formal context can be represented in a lattice, which is then called a *concept lattice*. There are various algorithms to efficiently produce concept lattices, but to understand the underlying principle a sketch of a naive approach is sufficient.

We know that for any concept (X,Y), we have $X = Y'$ and $Y = X'$ by definition. We can therefore rewrite any concept (X,Y) in two ways, that directly show a straightforward way of producing concept lattices. First of all, for any subset of objects X we have $Y = X'$. But then we can write $X = Y' = X''$, giving formal concept $(X, Y) \equiv (X'', X')$. Secondly, for any subset of attributes Y we have $X = Y'$ and with $Y = X' = Y''$ we thus get formal concept $(X, Y) \equiv (Y', Y'')$. In other words, to produce concept lattices we can take all subsets either of objects or attributes, and then apply the appropriate derivation operator as specified in the rewritten expressions. To summarize, a formal concept thus is a pair (X, Y) where each object in X has *all* attributes in Y, and where each attribute in Y is common to all objects in X. This relation between intent and extent of a concept is unique, so it does not fundamentally matter if you take subsets of objects or attributes as the point of departure. To clarify, let us discuss an easy example to illustrate this and also show sub- and superconcept relations.

3.2.1 Example of a formal context with visualized concept lattice

Ignatov provides a short example (see Ignatov, 2015, p. 7-8) that I reproduced here with my own Python code. We have four geometric objects and four possible attributes: they have either three or four vertices, a 90 degree angle, and can be equilateral. These attributes are either present or not, which gives the following formal context:

	3	4	90°	equil.
△	X			X
◺	X		X	
▭		X	X	
□		X	X	X

After applying FCA, we can visualize the concept lattice as such, where objects are labeled under the nodes and attributes on top of them:

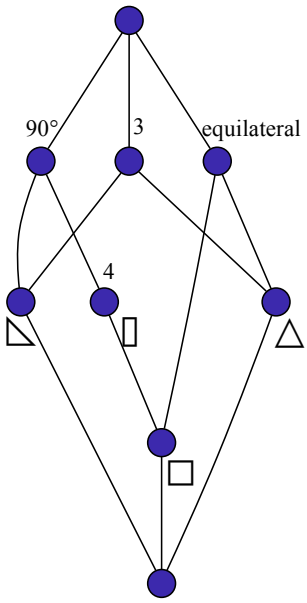


Figure 1: Simplified visualization of the concept lattice

Superconcepts are visualized higher in respect to their subconcepts. By definition, a concept lattice has a single highest concept that contains all objects but no attributes, and a single lowest concept that contains all attributes but no objects. At the lowest node, we have a concept $(\emptyset, A)$ with A the set of all attributes. Moving upwards, we incrementally increase the set of objects and try to find concepts. For example, when choosing the subset of objects that includes "square" we get the concept $(\{square\}, \{4, 90*, equil\})$. This is a formal concept because square has all those attributes, while simultaneously each of these attributes is common to *all* objects of the concept (which in this case, trivially, is only the square). We also see in the visualized concept lattice that "rectangle" is a superconcept of square. This *makes sense semantically*: a square is a type of rectangle with the restriction that it is equilateral. The concept "rectangle" within this formal context thus is $(\{square, rectangle\}, \{4, 90*\})$. Again, if we look at the formal context we see that all objects in the extension, i.e. both a square and a rectangle, have the properties that they have four sides and a 90 degree angle, and these properties are common to all objects of the concept. Because there is no

concept with an extension between $\{square\}$ and $\{square, rectangle\}$, the latter is the superconcept of the former, and the former is the subconcept of the latter.

Notice that the nodes are not labeled with the whole concept, which is a tuple of sets, but only partially through a particular reduction. Specifically, to prevent clutter and in order to highlight the main semantic structure, each concept can either be labeled by the extent or by the intent, due to the property that any concept (X, Y) can be rewritten as (X'', X') or as (Y', Y''). Labeling is further simplified in two ways:

1. either by only labeling with the object name that is not shared by subconcepts. E.g. we label $\{$*square, rectangle*$\}$ with "rectangle" because that is what semantically distinguishes this concept from its subconcept with extent $\{$*square*$\}$. Simpler said: we label this node with the rectangle because this is the first time it occurs from the bottom up.
2. or by only labeling with the attribute name that is common to all subconcepts but not to superconcepts, which occurs in the "higher" area of the concept lattice. E.g. the concept ($\{$*equilateral triangle, rectangle triangle*$\}$, $\{$*3*$\}$) is labeled with "3" because that attribute is the first occurrence of the common denominator of the two subconcepts ($\{$*rect. triangle*$\}$, $\{$*3, 90**$\}$) and ($\{$*equil. triangle*$\}$, $\{$*3, equil.*$\}$). Said simply: we label a node with an attribute the first time it occurs when moving top-down through the concept lattice.

These reductions effectively try to represent formal concepts as much as possible as lexical concepts without actually reducing information. In this manner, concept lattice visualization is a useful tool for highlighting semantic relations in a conceptual structure in a matter that is informative to humans.

3.3 Query by navigation

Based on the latter example we can also illustrate how query by navigation in such a conceptual structure works. If the extent of a concept contains documents and the intent some indication of shared content in those documents, then we can navigate through documents in two manners:

1. We can *expand our query* by "moving up" in the concept lattice, because a superconcept has a larger set of documents and a smaller intent. I.e. a higher concept describes more documents because these documents have to have less attributes in common.
2. We can *narrow our query* by "moving down" in the concept lattice, because subconcepts return less documents that have more attributes in common.

I will explain my implementation of query by navigation in a later section.

3.4 Interpretability

The aim of Formal Concept Analysis is to produce a structured representation of a space of objects and their properties. There are other methods available for achieving that goal. One could for example use clustering algorithms or even dimensionality reduction techniques such as SVD or PCA to extract features of a corpus of texts. When one however finds these features, e.g. clusters or principle component vectors, one invariably asks: but what do they really *mean*? In other words, after having done a data analysis and having some representation of the data, there remains an issue of the *interpretation* of that representation. Imagine we apply clustering to a corpus of texts. We will then always end up with a set of clusters, with texts within the same cluster being close to each other in terms of some distance measure. Under the assumption that texts that are close to each other in content represent a similar meaning, we then ask what that meaning exactly is? Does this cluster represent a concept? To complicate matters, we could have found different clusters had we used a

different measure for calculating distance between two clusters, or if we had changed the amount of clusters.

At this point, FCA really excels. There is no need for humans to for example think about what label to assign to a cluster, because a concept states a well-defined mathematical relation between objects and attributes, and we know of each of them what they mean. On top of that, the relations between different concepts are also exactly defined, so that for example we can specify for a given pair of concepts what their common superconcept is. There is no ambiguity in where one concept begins and another ends. Given some formal context, the outcome of FCA does not depend on the tuning of hyperparameters or a statistical computation that is only "understandable" by computers, because a concept expresses a direct relation within a formal context, without further abstractions. This does not mean that FCA presents a neutral representation of data, as if that would imply no interpretation at all. Decisions need to be made in determining how to represent the content of documents. For example, is every single term of a document a potential representation of the content of that document? Or can we select the most relevant terms? This thesis makes an effort to explain the choices in defining a formal context.

To summarize, FCA is a technique that facilitates the interpretation and understanding of data. This explainability is valuable for humans using intelligent computer systems, as humans are peculiar types of animals that always look for meaning. This thesis therefore pays special attention to interactive data exploration that keeps the human in the loop, presenting the user with information that is easy to understand, and that is based on concepts that can at any time be inspected and traced back to the documents they come from. Query by navigation is a good example of how a meaningful conceptual structure can support a user's search through documents in a transparent manner.

3.5 FCA in information retrieval

The particular issue that a company such as Wolters Kluwer deals with is one of *information retrieval* on a large scale, in this case for a corpus of specialized legal texts of about half a million documents. Currently, the search through this database is keyword-based. Queries are expanded based on a single hand-crafted synonym that is defined per document. This way of searching through documents is straightforward, but returns many results and requires the user of the system to smartly choose keywords in order to narrow down the scope of the search enough. Again, FCA can aid in this process by 1) creating a conceptual structure of the corpus itself, and 2) guide the user query by navigating through this structure.

Although the notion of concept is perhaps more intuitive when speaking about cats and their characteristics, the "formality" of a concept in FCA means that it is agnostic to the particular problem it is applied to. In the case of an information retrieval system, the extent of a concept is defined as a set of documents, and the intent is a set of document characteristics, for example important keywords. This then also directly shows the relevance of query by navigation. Moving up in the concept lattice expands the extent of the query, resulting more documents, whereas moving down in the concept lattice returns less documents. The latter is done by enlarging the intent, i.e. adding another keyword to the query. Based on the concept lattice, we know exactly which search term (part of the intent of subconcepts) we need to add to specify the query.

We have thereby reviewed the essential FCA terminology that is required for understanding this thesis. In the following section we will formulate the particular challenges of FCA in information retrieval applications and explain the approach this thesis takes.

3.6 Techniques for reducing concept lattice complexity

FCA as a technique has been around since the 80s, and by now is an established and "important formalism for knowledge representation, extraction and analysis" (Dias and Vieira, 2015, p. 7084). However, contemporary applications deal with increasingly larger datasets, and this makes FCA quickly prohibitive for many applications due to its high computational cost, which in the worst case is is $2^{min(|G|,|M|)}$ (Dias and Vieira, 2015, p. 7084). But apart from the computational cost of FCA, a main issue with larger datasets is that the resulting concept lattices quickly become too big, which entails that a concept lattice's "key aspects, those which are effectively sought, can be immersed in a maze of irrelevant details" (Dias and Vieira, 2015, 8074). In other words, this threatens the advantage of interpretability that I highlighted. "In fact, the problem of obtaining a concept lattice of appropriate complexity and size is one of the most important problems of FCA" (Dias and Vieira, 2015, 7084) and is so poignant for FCA because "the main applications make use of the concept lattice, usually represented by means of a line diagram, or a nested line diagram, or a tree diagram, etc." (Dias and Vieira, 2015, p. 7084). The latter visualizations can be very useful because they show a semantic conceptual structure that is readable and meaningful to humans. This desirable property is lost if conceptual lattices become too complex, in addition to full visualization becoming impossible, in the worst case.

I observe that while most publications of the technique of FCA itself are often relatively old, there are plenty of recent papers on reducing concept lattice complexity, further underlining the importance of dealing with complexity for the survivability of FCA as a relevant data analysis tool (e.g. Aswani Kumar and Srinivas, 2010; Buzmakov et al., 2015; Codocedo et al., 2011; Dias and Vieira, 2015, 2017; Medina, 2012; Sumangali and Aswani Kumar, 2018). The goal and challenge of such a complexity reduction is to simplify the concept lattice in such a way that the key aspects (i.e. the information of interest) are preserved. We therefore explore options to keep the desirable properties of FCA when applied to larger corpora, such as the legal database of Wolters Kluwer. Somehow the complexity of concept lattices needs to be reduced, while retaining the general desirable properties, in particular a clear enough semantical structure and support of query by navigation.

Dias and Vieira distinguish three classes of techniques used for concept lattice reduction (Dias and Vieira, 2015, 7085).

1. Removal of redundant information as much as possible, without changing the original structure of the concept lattice
2. Construct a high-level abstraction of the concept lattice that highlights the truly important aspects
3. Select formal concepts, objects or attributes based on a relevance criterion

In the context of information retrieval, the option of throwing away information is not desirable, as that excludes certain information from ever being found and by definition hurts the recall measure. Recall is important in the case of finding legal documents, as missing a relevant piece of information can have large professional consequences, and give an edge to opposing parties. However, removing redundant information is also not very suitable: "As redundant information removal techniques do not actually reduce the size of the lattice, they are not of much practical value" (Dias and Vieira, 2015, 7094). If it is a hard requirement that the structure of the reduced matrix is isomorphic, this option would be the best. However, to make it more feasible to use FCA for query by navigation in a way that presents the user with information of a sensible granularity, we really do want to reduce concept lattice complexity for our purpose.

For these reasons option 1 (redundant information removal) and option 3 (selection) are the least preferred potential methods for us. Dias and Vieira explicitly highlight the leftover option, simplification techniques, as "more suitable for applications that require interacting directly with the user", stat-

ing that they "have more applicability" than the other options (Dias and Vieira, 2015, 7095). They however also highlight an important trade-off for simplification techniques. On the one hand, simplification techniques are the most appropriate choice for handling "truly large formal contexts"(Dias and Vieira, 2015, 7095). But on the other hand, a strict isomorphism between the original and reduced concept lattice is lost and this could negatively impact the quality of the resulting concept lattice for some given application. The latter downside also makes this solution suboptimal for our purpose. If we first build a concept lattice and then simplify it, we might lose relevant information. Moreover, we do care about the structure of the concept lattice because we use it for query by navigation.

The important take-away here is that the main preference for simplification techniques would be 1) because of our focus on *applicability*, and 2) because of our focus on *scalability*. However, this thesis takes a different approach altogether to ensure applicability and scalability. The three classes of complexity reduction that are mentioned above all depart from a formal context that is already built. A fourth class, Dias and Vieira concede, would be a class of techniques that reduce complexity in some way *before* the formal context is built. This thesis explores such an approach in a pragmatic manner that fits the needs and *the existing infrastructure* of companies such as Wolters Kluwer.

3.7 Reducing complexity by leveraging existing search infrastructures

Companies dealing with big databases likely already have an infrastructure in place to search those databases, for example based on keywords. Wolters Kluwer offers their customers a search engine for finding relevant legal information from a large database of law textbooks, laws, case law, jurisprudence etc. In legal applications having a good recall is crucial, because missing relevant to a case might mean the difference between winning or losing that case. Ensuring good recall however comes at the cost of returning many documents, of which customers only have time to view some. This means that a further technique is needed to support the end user in finding what is relevant to him fast enough. I distinguish two classes of solutions:

1. Compute a relevance measure per document
2. Support the user in selecting relevant information more easily by him- or herself.

Even before looking at specific techniques, I propose to highlight on a more philosophical level a conceptual difference between these two approaches. One could say that computing a relevance measure is an "external" approach to understanding relevance, because it is based on an explicit computation over the contents of a document corpus, that ideally coincides with what is relevant to the user. The second approach does not in the same degree attempt to define objectively what is relevant to the user, but instead lets the user act based on his or hers subjective "internal" sense of relevance. In that case, a main issue is not whether the "explicit" computation of relevance coincides with what the user deems relevant, but instead to provide the user with enough information to enact his or her own sense of relevance. In other words, when speaking about relevance we can consider objective facts concerning the corpus (e.g. term frequency), but also in a more subjective sense what corresponds to the preferences of the user.

A clear advantage of the first approach is that the user is immediately presented with the results that are deemed most relevant, which reduces effort on the side of the end user. Various approaches to computing relevance exist. One could for example compute TF*IDF scores for terms from a user query to get an indication of how relevant that query term is in a particular document. More advanced approaches could take into account client profiles to tailor the search results to a specific user, although it is not clear how accurate using "client profiles" would be here, because the legal professionals that use the search service do not represent themselves but others, in cases that all have their own context. In a sense, for personalization of the results one would thus need "case profiles" instead. But in all cases, for big databases this would require a Big Data solution that is currently not realistically feasible

for many companies, as it would usually require a big investment in changing ICT infrastructures and software.

The second approach reserves a more active role for the user to select relevant results. From a conversation with an IT employee from Wolters Kluwer, it became clear that Wolters Kluwer has ambitions to provide techniques that follow this line of thought. One could for example think of automatically highlighting and presenting important blocks of texts, such as the particular charge that instigated a legal case or the final ruling of the judge, so that it becomes quicker for the end user to inspect these texts on relevance. The "query by navigation" approach this thesis takes also belongs to this category. I propose an implementation that presents the user with meaningful choices that support finding relevant results *interactively*. This gives the end user a lot of freedom and control over refining or broadening queries, while avoiding that the user has to deal with a bulk of individual texts. Instead, the user interacts with a *meaningful representation* of a group of texts, namely a concept lattice that is formed based on an analysis of the contents of the documents.

However, in order for the query by navigation approach to actually support the user in finding relevant results quickly, and if this approach should not be too complex, then we again quickly return to the issue of FCA's computational complexity. Computing a formal context over all texts in a large database is computationally infeasible, and to keep navigation user-friendly and efficient a lattice needs to be small. In line with the fourth class of lattice complexity reduction, as mentioned above, I propose a method that extends traditional search with FCA and query by navigation, as follows:

1. A subset of documents is returned in a traditional search query
2. FCA is applied on this smaller subset of documents to build a concept lattice
3. This allows query by navigation to further refine the search interactively

So we reduce the complexity of our concept lattice before the creation of any formal context. The underlying rationale is that in this two-stage process, the first stage is coarse-grained and efficient, so that a fine-grained approach based on a semantic analysis of the documents themselves becomes feasible in the second stage. *We choose the right granularity at the right time.* Another advantage of this approach is that the recall and accuracy of the process are the same as for the chosen search method, because a full Formal Concept Analysis can now be applied without losing any information.

To summarize, this approach is taken because it uses both above-mentioned approaches to relevance at the appropriate time. It narrows down the set of documents quickly using some search method, that might employ and "external" relevance calculation, after which the user can further navigate based on what in the conceptual structure is relevant to him or her ("internal" relevance). Furthermore, this approach is very practical for companies because it is an extension to existing search mechanisms and infrastructure rather than a replacement of them.

Our goals were to make FCA *applicable* and *scalable*. Applicability is ensured by performing 1) query by navigation, which is a clear application with the user in mind, and 2) by not performing query by navigation on *all* texts, which would be too laborous for the user. Scalability is better by only applying FCA when sufficiently zoomed in on a relevant subset of texts. Moreover, Wolters Kluwer does not deal with streaming data, so pre-computation of formal contexts is possible if the set of possible queries is known in advance. This should be possible for Wolters Kluwer, because they maintain a thesaurus with usable search keywords that describe the legal corpus. If new texts matching a particular keyword are added, only the formal context of documents resulting from a search on that keyword have to be recomputed. Scalability is now of course co-determined by the chosen search method. But the approach put forward here is agnostic with respect to the particular search method used, and it is assumed that in any real scenario where this approach is potentially applied, the pre-existing search mechanisms are already feasible - otherwise they would not be used.

With these goals in mind, and based on the given justifications, I implemented a data pipeline that

starts at raw xml files in the database of Wolters Kluwer, and ends in a query by navigation application. In the following section I explain choices made in the implementation of this pipeline.

4 Data pipeline: Design choices

Within this thesis a data pipeline was designed that starts with raw data and ends with a query by navigation prototype based on fCA. In the following section I will provide an overview of this pipeline, with special focus on some design decisions that were made and their justification.

4.1 Overview data pipeline

This thesis implements a pipeline with the following components.

1. Simple search over corpus for documents matching a keywords from the Wolter Kluwers thesaurus. Can be replaced by any other search method.
2. Extracting relevant text by parsing the remaining xml documents
3. FCA: Creation of concept lattice
   (a) keyword extraction with lemmatization and PoS-tagging
   (b) attribute determination using stopword removal and relevance calculations over keywords
   (c) computing of formal context by checking attribute presence per document
   (d) computation of concept lattice with Formal Concept Analysis
4. Query by navigation over concept lattice

Figure 2 shows a graphical overview of the proposed pipeline.

4.2 Design choices

Several components of the general outline above demand an explicit interpretation in order to be implemented. Such an interpretation is a design choice that needs to be highlighted and justified. The following interpretative issues need to be answered in practice:

1. How do we define search over the Wolters Kluwer corpus?
2. What do we consider relevant text?
3. What is a keyword exactly?
4. What is an attribute in our formal context?
5. What do we present to the user in order to navigate?

I will discuss relevant design decisions concerning these points in order.

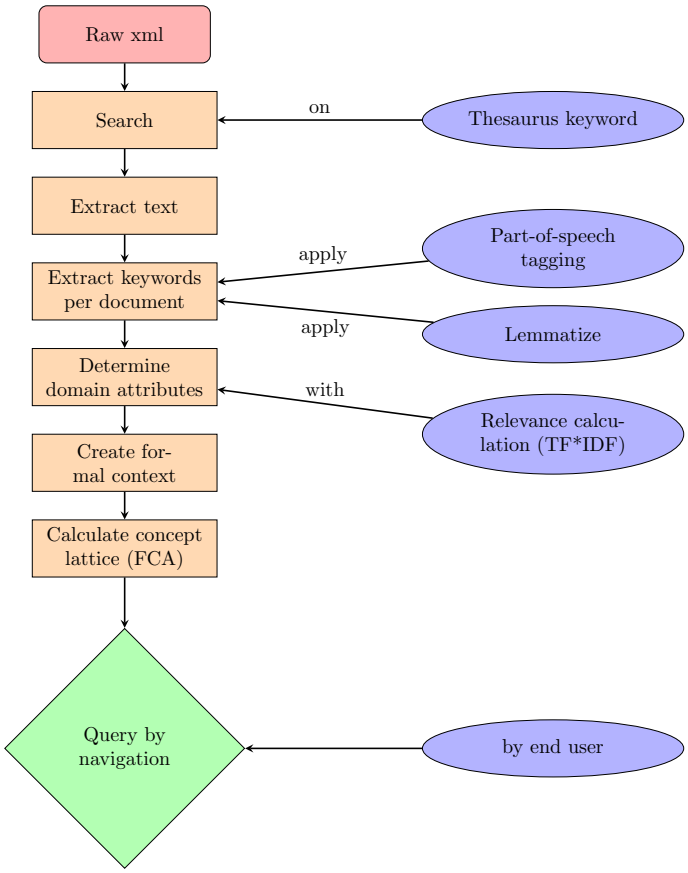


Figure 2: Proposed pipeline

4.2.1 Search

The work in this thesis is presented as a possible extension to existing search systems, but it nevertheless is agnostic with respect to how this search system is implemented or how advanced it is. For a proof of concept, the main requirement for a search is that

1. it returns a sufficiently small subset of documents
2. that have a minimal relevance guarantee for the given query

This is guaranteed in the current pipeline. Documents of the Wolters Kluwer database are annotated by hand by a search term. It is safe to assume that this manual labeling by experts ensures minimal relevance of documents if their search term matches the user query (requirement 2). The Wolters Kluwer thesaurus contains many of these search terms that are sufficiently specific to return relatively small subsets of documents to which FCA can be applied. For example, a search on a quite specific term such as "beroepsopleiding" (professional education) returns 38 documents, whereas a search on one of the most general terms possible such as "arbeidsrecht" (labour law), which effectively describes a whole domain of law, returns 624 documents. (Note that "arbeidsrecht" here is a search term belonging to the category "Arbeidsrecht / Algemeen" and not an identifier for the whole field of "arbeidsrecht").

For the purpose of a proof of concept within the scope of this thesis, a very simple search based on these search terms of the thesaurus thus suffices. Because of the *modular* pipeline approach, search

can be interchanged with another search method without affecting the rest of the pipeline.

4.2.2 Extract Relevant Text

In order to build a formal context for the purpose of information retrieval, we need a set of features ("attributes" in FCA lingo) that are able to sufficiently represent the content of documents. A first question is where to look for these attributes. Based on inspection of the xml-markup of documents in the Wolters Kluwer corpus, I have identified the xml tags in which the "content" of a document is displayed. The data pipeline inspects the contents of both a search summary and the article title and body, because these are considered to contain appropriate representations of the document content. Text that I considered inessential, such as footnotes, which contain technicalities, formalities and references, and xml-metadata, are discarded.

4.2.3 Keyword extraction

After selecting text that contains a document's content, we need to represent this content in a manner that aids the construction of a formal context.

The approach of this thesis is to represent document features as keywords (1-grams, specifically). If the chosen methods in the prototype pipeline work well, extending this representation to bi-grams is an obvious candidate for follow-up experiments. However, especially because the complexity of the techniques explored in this thesis are of critical importance, it makes sense to start with a good baseline representation that does not impose extra complexity challenges. Furthermore, although the final keywords are 1-grams, word context is nevertheless taken into account because a preliminary selection of descriptive keywords is done based on PoS-tagging.

There are two more important choices to be made about what type of keyword we consider to describe a document feature. Two requirements need to be taken into account:

1. Different inflections of the same word need to mapped onto the same keyword.
2. A pre-selection needs to be made about what type of word is descriptive of a document's content.

Firstly, different inflections of the same word have a different morphological form but are semantically intertwined. From a practical perspective, defining each single word as a keyword would lead to an extremely large set of keywords. Since our document features are supposed to represent the semantical content of documents, it is therefore desirable to map different inflections onto the same keyword. To achieve this we could for example reduce words to their stem, or lemmatize them. However, inflections of different lemmas that really have different meanings could end up with the same stem, which makes stemming undesirable for our use case. Moreover, because our goal is to define attributes to be used in FCA and in a user-facing query by navigation application, it is a must that the resulting keywords are clearly readable and interpretable by the end user. This also justifies a choice for lemmatization, as this results in proper words that can be found in a dictionary, whereas this does not hold for stems. In this manner, the user can be prompted with meaningful well-formed words in the query by navigation prototype.

Secondly, many words, e.g. stop-words, are clearly not descriptive of any content. The approach chosen for this thesis is to tokenize and PoS-tag words (Part of Speech), and only nominate words with particular tags that are deemed suitable as a feature description. Much readily available software supports processing the English language. If there is support for Dutch, the trained models are often not of production quality. However, there is a software package called Frog that offers high quality natural language processing for Dutch (Van den Bosch et al., 2007):

> Frog's current version will tokenize, tag, lemmatize, and morphologically segment word tokens in Dutch text files, will assign a dependency graph to each sentence, will identify the base phrase chunks in the sentence, and will attempt to find and label all named entities. (Van der Sloot et al.)

So we can use Frog to tokenize, PoS-tag and lemmatize Dutch texts, which is all we need based on the above reasoning. Frog uses the CGN tagset ("Corpus Gesproken Nederlands") for Dutch PoS-tagging, which is extensively documented and justified (Eynde, 2004). A Frog client in Python has been written by Maarten van Gompel, who also maintains the software distribution LaMachine, which contains Frog and all its dependencies and is available under the GNU General Public License (Van Gompel).

Part of Speech Tags used by Frog The CGN set of PoS-tags contains a grand total of 320 tags, divided over various categories (Eynde, 2004). Notably, because Frog also recognizes interpuction (LET()) or special tokens, such as interrupted words (SPEC(afgebr): e.g. "binnen-"), or other languages (SPEC(vreemd): e.g. "wishful", "ad hoc"), considerably less manual preprocessing was necessary to get sensible results.

Some categories of tags can be straightforwardly excluded as candidates for keywords, such as "telwoorden"(cardinal numbers), "voornaamwoorden" (pronouns), "lidwoorden" (articles), "voorzetsels" (prepositions), "voegwoorden" (conjunctions), "bijwoorden" (adverbs), "tussenwerpsels" (interjections), special tokens and interpunction.

A candidate that is left is the category of "adjectieven" (adjectives). Examples from this category are: "bester kwaliteit", "het leukste is dat", "zaliger gedachtenis", "het is hier stilletjes", "het langst slapen". Another candidate is the category of "werkwoorden" (verbs). Examples are: "zal komen", "een getemde feeks", "het geschrevene", "liep lachend weg". The last candidates are nouns. Examples are: "deze muziek", "dit stoeltje", "de Ardennen", "het Nederlands" "een riool", "Linux" (Eynde, 2004, 62-74).

What distinguishes noun-tags from adjective- and verb-tags is that nouns indicate some (real-life) object, whereas adjectives offer further qualifications of these objects, and verbs indicate some relation of an object. If in a formal context our objects correspond to nouns, then adjectives and verbs could be meaningful attributes. However, we define a formal context on a higher level of abstraction, where objects correspond to documents. In that context, attributes can be thought of as what these documents "are about", which roughly corresponds to the entities indicated by noun-type phrases. Because adjectives and nouns are mostly meaningful in relation to those nouns rather that the objects (i.e. documents) themselves, and because we are trying to define attributes *of those documents*, it is justifiable to single out words corresponding to noun-tags as potential attributes for our formal context. Even though in some special instances words of other tag categories might result in possible keywords, it is not desirable to seek this level of nuance because the main priority is in fact to reduce the amount of attributes for our formal context as much as possible, to limit the combinatorial explosion that comes with FCA of a large formal context.

After producing a list of keywords per document, it needs to be decided which keywords are relevant for the *whole corpus of documents* under consideration. In other words, if we use FCA to navigate over a corpus, which keywords are appropriate attributes for the required formal context?

4.2.4 Formal Context Attributes: Domain feature extraction

After the previous step we have a set of keywords *per document*. In order to perform FCA on a set of texts we however need a set of attributes that is able to describe the contents of all documents in a corpus, i.e. we need *domain attributes* describing the subset of documents returned by a query. But

when we simply add up the keywords of individual texts and use those as domain attributes, we end up with thousands of attributes in our formal context, even after the previous preprocessing steps. But this again would make the calculation of a concept lattice prohibitive very quickly. We have limited control over the amount of objects in the formal context because we do not know in advance how many documents a query will return, but we can choose the amount of attributes we use. If we however manually limit the amount of domain attributes we choose from all available keywords, we need to ensure we pick the most relevant ones. This step is non-trivial, and this thesis will compare three different methods for calculating relevance.

Domain specific keyword removal Before calculating relevance, I found it beneficial to remove a hand-selected set of keywords specific to the legal domain. Both the previous preprocessing steps and the following TF*IDF relevance calculation minimize the influence of stop words, but nevertheless I found that a small set of lemmas were so frequent that they were almost always picked as most relevant, irregardless of the relevance measure. They are however often not informative, because they pertain to irrelevant formalities of the legal domain. I selected the following "legal stop words" to removed from consideration as a formal context attribute: {advocaat, advocatuur, artikel, auteur, appellant, deel, eis, geval, gerecht, gerechtshof, hof, jaar, jurist, lid, punt, raad, recht, rechter, rechtbank, verweerder, wet}.

Approach 1: Term frequency (TF) A first straightforward option is to take term frequency as a indication of relevance. Usually, term frequency is computed within each document, because this allows the ranking of documents in a query, based on how important a query term is in any given document. In our case, we can extend this approach to the domain under consideration by calculating a frequency distribution over all words in the corpus that we apply FCA to, and then taking the top n most frequent keywords as domain attributes. The rationale is that if a keyword appears in most documents of a particular corpus, then it must be relevant not only for a single document but for the corpus as a whole.

An obvious drawback of this approach is that frequent but unimportant words, i.e. stop words such as "the", are then unjustly regarded as important. This makes preprocessing steps, such as stop word removal, absolutely necessary in this approach. Note that the approach with Frog taken at the previous step, effectively achieves this by only returning words belonging to noun phrases as keyword candidates.

Approach 2: Term Frequency x Inverse Document Frequency (TF*IDF, with a note on complexity) The simplicity of using term frequency is both its charm and its downside. Term frequency is only a reasonable indication of relevance if the frequent term is not a stop word or some other filler word.

Although stop words in the ordinary sense of the word are filtered out at the lemmatization step, this issue of the term frequency measure still remains when describing domain attributes. For example, when we search on the term "beroepsopleiding" the word "advocaat" (lawyer) is very frequent. According to the term frequency measure, this term is then also a highly relevant descriptor of the group of documents we retrieved by searching on "beroepsopleiding". But for any other query than "beroepsopleiding", it is very likely that "advocaat" will also occur very frequently. If this is the case, then this term does not distinguish the results of different queries well, from which we can conclude that it is not specific enough to a particular domain in order to capture what distinguishes that domain from others. This issue is especially expected to become prominent if the search returns larger groups of documents. It is very well possible that then the top n most frequent terms are all platitudes, such as "lawyer". As a result, during query by navigation the user will then not be able to narrow the query down to a small group of documents fast enough, because all attributes appear in many documents.

A common technique to counter this problem is called TF*IDF (Spark Jones, 1972), which multiplies the (normalized) term frequency (TF) of some term with the inverse document frequency (IDF). The rationale behind TF*IDF is that a frequent term of some document that occurs *in many documents* is more likely a stop word and less distinguishing than a term that is very frequent in a particular document but *appears in relatively little other documents*. In other words, if a term occurs in almost all documents of a given corpus, its document frequency (DF) is very high, and its inverse document frequency (IDF) is thus very low, resulting in a low TF*IDF relevancy score, despite perhaps having a very high term frequency (TF). In this thesis, smoothening is applied to the IDF computation to avoid divisions by zero. The inverse document frequency with smoothening is computed as such, where DF is the number of document in which term t appears (Pedregosa et al., 2011):

$$IDF(d,t) = \frac{log((1+n)}{(1+DF(d,t))) + 1}$$

This technique is usually used to offer a document ranking for a given query. Given some query, the TF*IDF scores of terms in that query can be computed per document. Documents with higher TF*IDF scores are, according to this measure, considered to be more relevant and will rank higher in the search results. The problem formulated in this thesis is slightly different. Instead of computing the relevance of documents given a query directly, the goal instead is to extract *relevant domain attributes* for use with FCA. If we want to delimit the complexity of FCA through reducing the amount of used attributes in the formal context, then it is very important to choose the most relevant attributes in order not to lose too much expressivity. So how can we use the rationale of TF*IDF to get a better solution to finding *domain attributes* rather than relevant terms for a single document?

This is the approach taken:

1. Take the list of lemmas extracted from the document subset *with duplicates* as potential domain attributes.
2. Compute the normalized term frequency of these lemmas
3. Scale these term frequencies with the IDF scores of these lemmas, computed over the bigger corpus.

In other words, we act as if the selected lemmas of the documents returned by a search compose one document. By then calculating the TF*IDF score per lemma to produce a ranking over all lemmas within one document, rather than over one lemma in different documents. The top n lemmas are considered the most relevant, and are chosen as attributes for our formal context.

It is of crucial importance that the IDF is computed over a way larger corpus than the set of documents of which we extracted the lemmas. Imagine we would compute the IDF over the same corpus as we compute TF over, for example for the query "beroepsopleiding". It is then rather obvious that the lemma "beroepsopleiding" would appear almost in every document. In terms of FCA, it would be meaningful if "beroepsopleiding" would characterize very general concepts, that we can further specify with query by navigation. However, if we compute the IDF scores over this smaller set of documents, "beroepsopleiding" will be heavily penalized because it occurs in many documents. As a result, this potentially relevant lemma that describes this group of documents will be unfairly downvoted in our relevance ranking.

If instead we use IDF scores computed over the larger corpus for TF*IDF, beroepsopleiding is expected to still be ranked relatively high, provided that "beroepsopleiding" is a lemma that does not appear very frequently in all other documents of the larger corpus. Therefore, the inverse document frequency needs to be based on a sufficiently large corpus so as to not be biased by a particular subset of documents or domain. Ideally, this means computing the inverse document frequency over the whole corpus (500.000 documents). The morale of this thesis is that by making more intelligent choices

in the preprocessing steps for determining attribute candidates, i.e. by lemmatizing and making a deliberate selection of appropriate lemmas, the complexity of the formal concept analysis is reduced. To keep all results comparable, the same preprocessing steps to select attributes must be performed as preparation for computing TF and IDF scores. This use of lemmatization does, of course, increase the computational load of the preprocessing steps, which makes computing the inverse document frequency over the whole corpus time intensive. More importantly, for fault tolerance the resulting lemma's have to be written to disk, to avoid possible recomputation on failure. Even though Frog's lemmatization is itself fast with a rate of 900 words per second according to Van der Sloot et al. , writing to disk is notoriously slow. On the machine used for this thesis (which is shared with others), a worst case time for lemmatizing a long text and writing it to disk was close to 2 seconds. If we assume this worst case scenario in which every document is very long, then for 600.000 documents this computation would take $\approx$ 14 days. After that, the IDF scores over all terms still need to be computed.

In production settings, with a professional hardware setup, doing those computations is not an issue though and moreover, recomputation of the IDF would be very infrequent given that IDF values are stable when computed over a large corpus. For this thesis, the IDF scores were computed over 143.275 documents, which is sufficiently large for the IDF scores to not be biased, considering that the size we amount of documents we compute term frequencies over is more in the range of 0-500 documents. This computation took little over two days. An interesting observation here is that, in a meeting with an ICT officer of Wolters Kluwer, they stated that they once tried to compute TF*IDF scores over the whole dataset, but that this attempt failed due to a too high computational load. In the approach taken in this thesis, however, computing the IDF scores over the whole corpus is feasible with more computing power, because even though IDF scores are then computed over all documents, they are *not* computed *over all terms*, but over selected lemmas. There are less lemmas due to the fact that similar terms are mapped on the same lemma, and also because of further informed selection of lemmas. In the proposed pipeline we can make such informed decisions because we have a particular application, FCA, in mind for which this choice is appropriate. A similar choice might however not be the most appropriate for different applications.

Approach 3: IDF It might however also be interesting to disregard term frequency altogether, and look only at the IDF values of terms. This could potentially have benefits. Whereas using term frequency, even scaled as TF*IDF, primarly defines the attributes of a set of documents as what they have in common, using only IDF would instead represents the attributes of this set of documents as what is most unique and distinguishing compared to other documents. This would lead to attributes that are extremely specific to this set of documents, which could be appropriate because we are after formulating domain attributes after all. Although TF*IDF is the most common approach, using only IDF could be an interesting option for the aims of this thesis for three reasons.

First of all, if we put a hard limit on the amount of attributes we will use in a formal context for the purpose of reducing the complexity of the analysis, then we might want to prioritize the most distinguishing attributes. A risk of using term frequency in any way is that the top n terms will be platitudes such as "lawyer". This might be appropriate for a conceptual representation of what these texts are about, but less for information retrieval.

Secondly, because we know we are working with specialized texts from the legal domain, it may be more interesting to select a relevance method that is sensitive to the very specific and specialized terms of legal jargon.

Thirdly, if the resulting attributes are indeed more distinguishing between documents, the formal context will be very sparse. As a result, the resulting concept lattice is expected to be much smaller and shallow. This would greatly lower the computational load of doing Formal Concept Analysis, but for our purpose would have the added benefit of allowing faster query by navigation. This is because

in this scenario a single decision for query expansion will single out more of the other documents on account of being very specific, thus taking less steps to reach the bottom of the concept lattice through navigation. Consequently, it is expected to take less steps to single out a very select subset of documents, or to say it simply: a lawyer can potentially zoom in on his particular interest faster.

There might be an interesing tradeoff between the potential benefits of a sparse formal context with very distinguishing attributes, and whether these attributes are still sufficiently meaningful to users or too detailistic.

4.2.5 Query by Navigation

Query by navigation is implemented on top of a concept lattice. Navigation is thus not just simply navigating a hierarchical tree structure, but is *conceptual navigation* and says something about what a group of documents means. Both the intent and extent of concepts grow big in information retrieval tasks, and even though conceptual relations indicate semantic differences between texts, they are not always necessarily easy to interpret for humans. When implementing query by conceptual navigation we thus need to think about how to provide information to the user about which concepts can be explored in the concept lattice, depending on the current concept in the navigation process. It should be easy for the user to understand the meaning of the possible choices. This is achieved in two ways.

1. The objects of the formal context are document IDs, which are not meaningful in themselves for users, even though it is perfectly possible to navigate based on increasing or narrowing the extent. The attributes of the formal context are however understandable in natural language for the user. Therefore, navigation will be performed by presenting choices for widening or narrowing the *intent*.
2. It is not user-friendly to display the whole intent of concepts that can be visited. Instead, the user is presented with the *set difference* between the intent of the current concept and of each concept that can be visited. Often that means that the user is presented with a single attribute in natural language, which is easy to understand, but if the set difference is several attributes, all of these will be displayed together. The intent of the current concept is always visible in full to the user, so the user can at all times access all information about the intent of each candidate concept by taking that extent and adding the displayed set difference.

Further remarks:

- Because navigation is achieved by increasing the intent of concepts, the concept lattice is navigated from top to bottom, where the "highest" concept (the lattice supremum) has all documents as extent, and an empty set as intent. By specifying more attributes, the extent diminishes until we ultimately reach the "lowest" concept (the lattice infemum) that has an empty set as extent and an intent with all attributes.
- Choosing an initial concept as a starting point of the navigation process can be done both by querying on one or more document IDs, or based on one or more attributes. If an exact match is not found, the closest matching extent or intent (depending on what you provided) is taken as the initial starting point. If no arguments are provided for the query, the query starts with the lattice's supremum.
- Because of the top-down approach there is currently not a function to directly move back up the lattice in a way that retraces the path taken. Instead, this functionality is effectively achieved by querying on the concept you want to backtrack to.
- The user is thus not displayed with a static list of attributes, but with a dynamic list of attributes that depends on the lower neighbouring concepts in the concept lattice.

- At any time the extent can be shown, and there is a command available to the user to inspect any document in the extent of the current concept.

5 Results

In this section I provide results for a number of example search queries. I will report on

1. Statistics of set of documents returned by the search
2. Attributes of the formal context and their properties
3. Properties of the resulting concept lattices

Evaluation of concept taxonomies, whether it is done manually or automatically, is in the scientific literature achieved by comparison with so-called golden-standard ontologies. However, such an ontology is not available for the Wolters Kluwer database. Neither does it make sense to evaluate performance in terms of accuracy or recall, because in the current pipeline those are the same as the search method used in the chosen hybrid approach. In the following I will instead provide an in-depth manual review of five examplary results produced by the implemented pipeline.

5.1 Query results

Example queries are taken from different main topics from the Wolters Kluwer thesaurus. They are sorted in increasing order on the amount of documents that they return.

Table 1: **Seksuele geaardheid**

Rubric	Burgerlijk recht
Main topic	Arbeidsrecht
Subtopic	Arbeidsrecht / Algemeen
Keyword	Seksuele geaardheid
Number of documents	25
Number of attributes	1145
Keyword lexical diversity	0.08
Attributes / Document	45.80

Table 2: **Beroepsopleiding**

Rubric	Recht algemeen
Main topic	Juridische beroepen
Subtopic	Juridische beroepen / Notaris
Keyword	Beroepsopleiding
Number of documents	38
Number of attributes	1035
Keyword lexical diversity	0.32
Attributes / Document	27.24

Table 3: **Automatisering**

Rubric	Ondernemingsrecht
Main topic	Informatierecht
Subtopic	Informatierecht / ICT
Keyword	Automatisering
Number of documents	130
Number of attributes	4084
Keyword lexical diversity	0.20
Attributes / Document	31.41

Table 4: **Terrorisme**

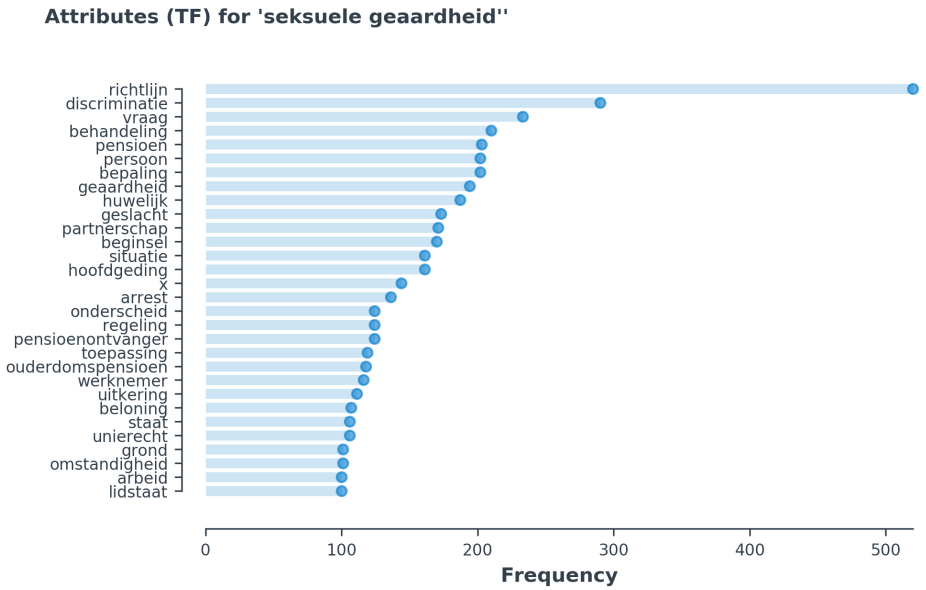
Rubric	Openbaar bestuur
Main topic	Openbare orde en veiligheid
Subtopic	Openbare orde en veiligheid / Terrorismebestrijding
Keyword	Terrorisme
Number of documents	247
Number of attributes	4965
Keyword lexical diversity	0.10
Attributes / Document	20.10

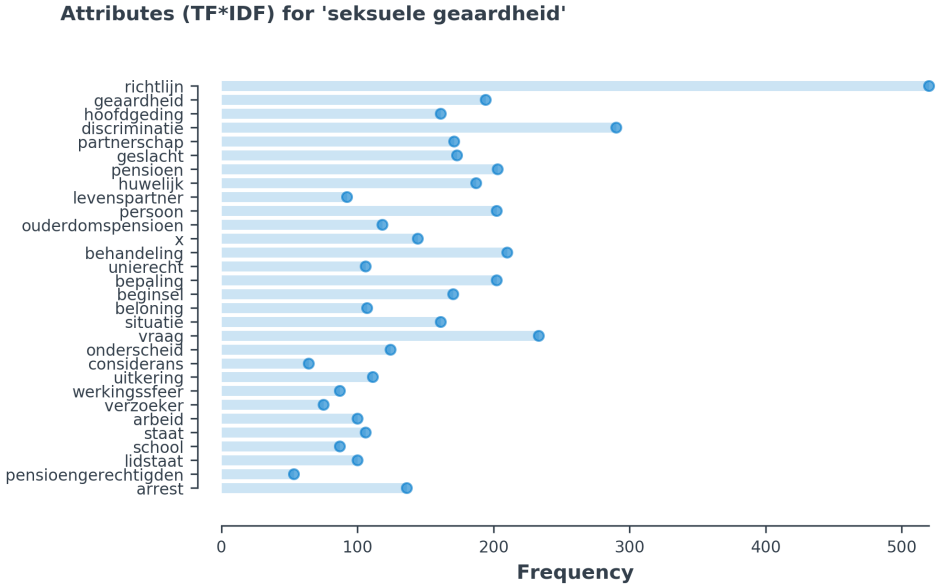
Table 5: **Arbeidsrecht**

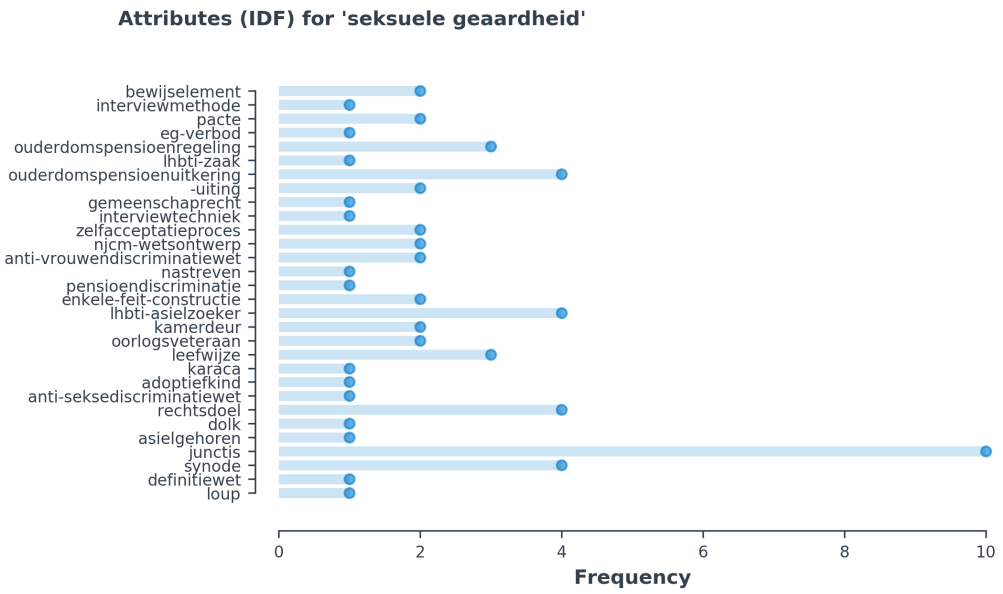
Rubric	Burderlijk recht
Main topic	Openbare orde en veiligheid
Subtopic	Arbeidsrecht / Algemeen
	International privaatrecht / Internationaal bevoegdheidsrecht
Keyword	Terrorisme
Number of documents	624
Number of attributes	9548
Keyword lexical diversity	0.07
Attributes / Document	15.30

5.2 Formal context attributes for different relevancy measures

5.2.1 Seksuele geaardheid (n=25)







The difference between TF-attributes and TF*IDF-attributes is only six attributes and thereby relatively small. The TF-attributes { grond, omstandigheid, regeling, toepassing } are all very general and not particularly informative. They are replaced through TF*IDF by {considerans, levenspartner, school, verzoeker}. “Levenspartner” and ”School” have a clear relevance for the query. “Considerans” and ”Verzoeker” indicate an aspect and a role in a legal process, and are not much more informative than the replaced TF-attributes. Two other TF-attributes, {pensioenontvanger, werknemer} are replaced by very similar terms that are apparently more rare in the overall corpus, {pensioengerechtigden, werkingssfeer}.

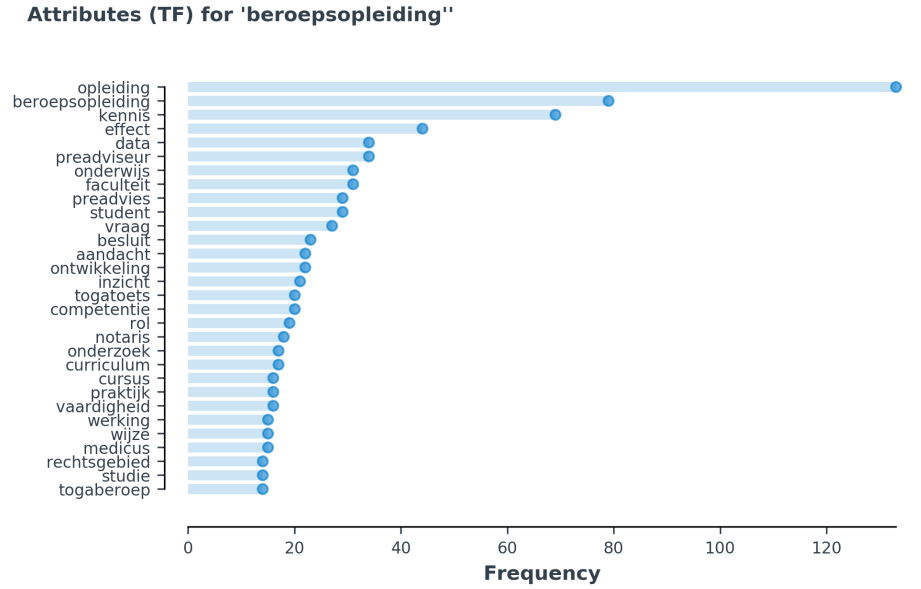
The small difference between TF- and TF*IDF attributes is to be expected because this query returned a small amount of documents. In general the selected attributes seem relevant for the query term. The attributes {behandeling, discriminatie, geaardheid, geslacht, huwelijk, levenspartner, onderscheid, ouderdomspensioen, partnerschap, pensioen, pensioengerechtigden, werkingssfeer} all have a relevance that is quite easily understood. Some attributes refer more to formal aspects of the legal documents, such as {arrest, beginsel, bepaling, considerans, hoofdgeding, richtlijn, verzoeker}. This does not necessarily make them irrelevant though, because for lawyers these terms are meaningful. An “arrest” is for example stated in higher court, and says something about the type of legal document that can be retrieved.

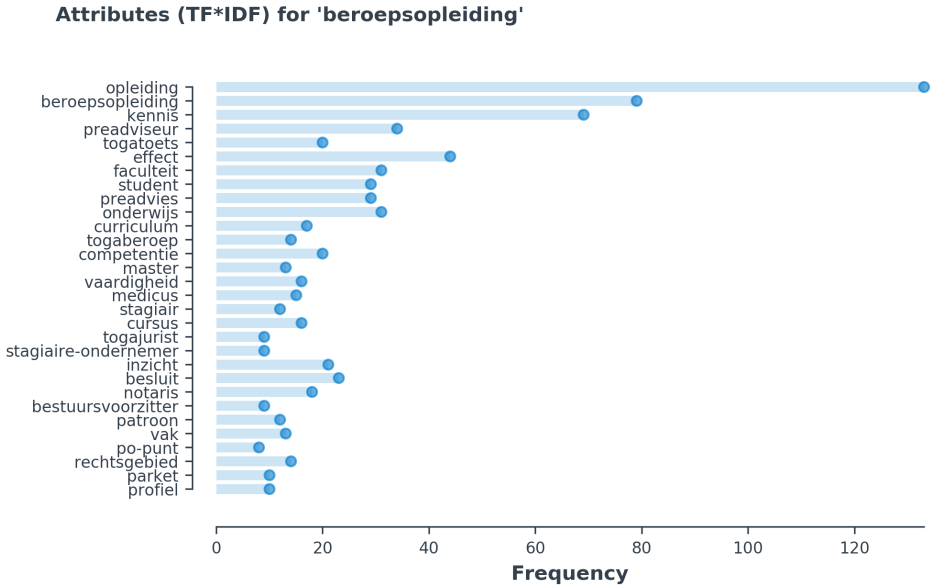
Some very general and seemingly useless terms that remain both in TF and TF*IDF are {persoon, situatie, vraag}.

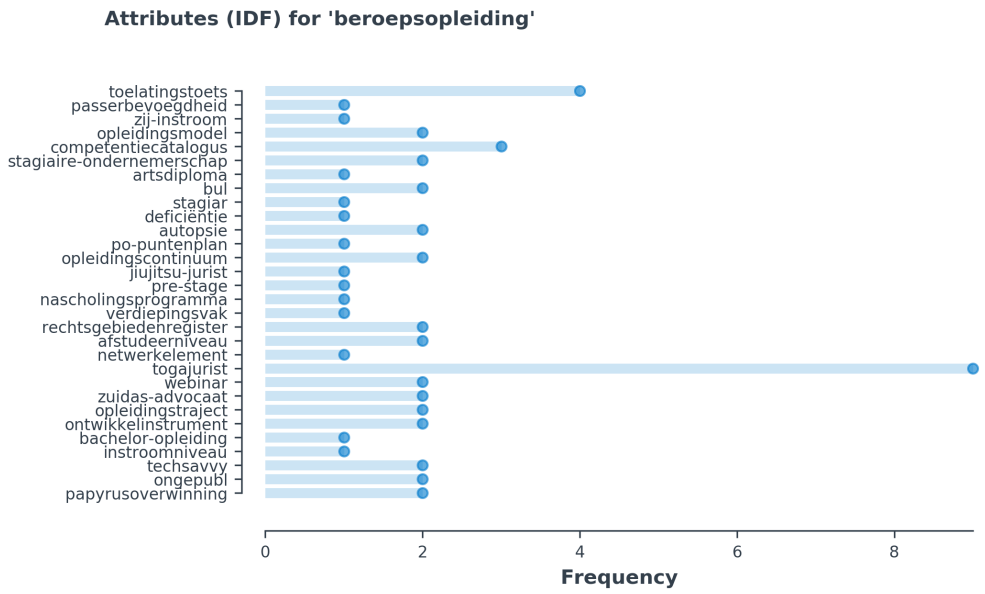
An interesting outlier is the term “X”. If we query on X using query by navigation and inspect the document in which it occurs, we see that “X” is a placeholder for persons, that due to the sensitive nature of the subject of sexual orientation do not appear in the document by name.

In the IDF-attributes we see a few attributes that fit the query term very well, such as {lhbti-zaak, zelfacceptatieprocess, anti-vrouwendiscriminatiewet, lhbti-asielzoeker, anti-seksediscriminatiewet}. “lhbti” stands for ”lesbische vrouwen, homoseksuele mannen, biseksuelen, transgender- en intersekse personen.” We can also see that “njcm” is an interesting term here, because that stands for ”Nederlands Juristen Comité voor de Mensenrechten”, and thus concerns human rights. Many of the other attributes are very specific jargon. Most terms are not very relevant, and from the counts in the IDF graph you can see they are so specific that they only occur in one or two documents.

5.2.2 Beroepsopleiding (n=38)





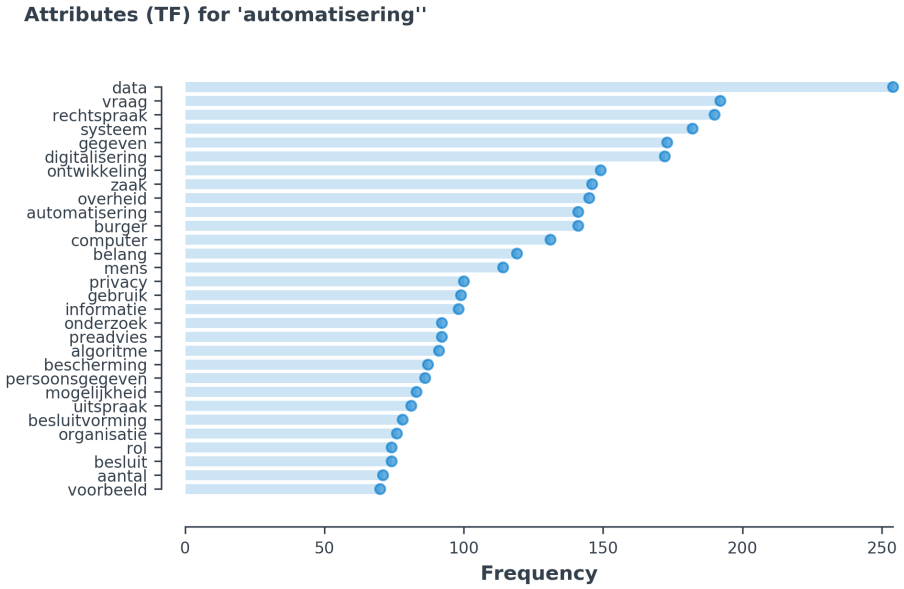


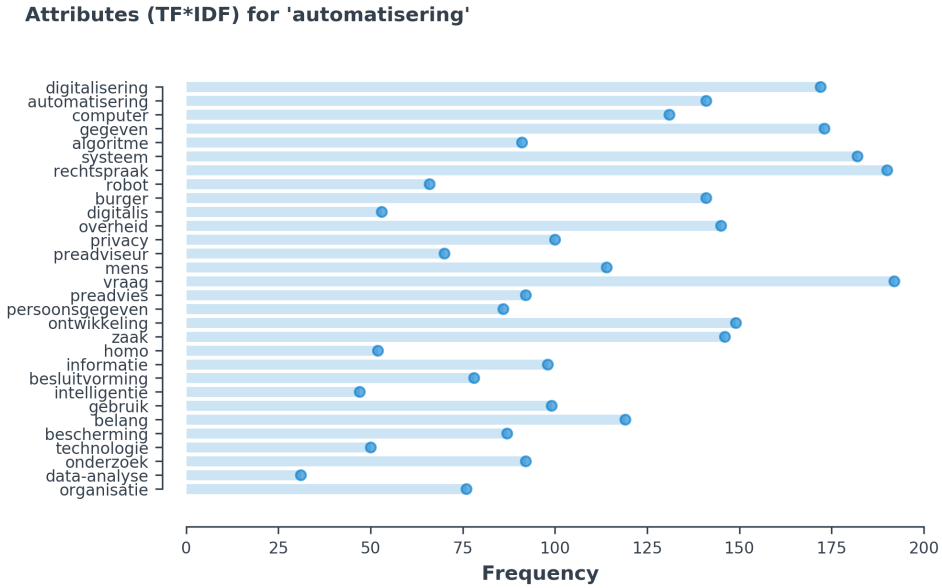
A look at the attributes in all graphs shows that almost all attributes fit the query "beroepsopleiding" very well. It is good to realize that this set of documents is focused on the education of legal professionals in specific. The difference between the TF- and the TF*IDF-attributes are that the TF-attributes {aandacht, data, onderzoek, ontwikkeling, praktijk, rol, studie, vraag, werking, wijze} are replaced in TF*IDF with {bestuursvoorzitter, master, parket, patroon, po-punt, profiel, stagiair, stagiaire-ondernemer, togajurist, vak}.

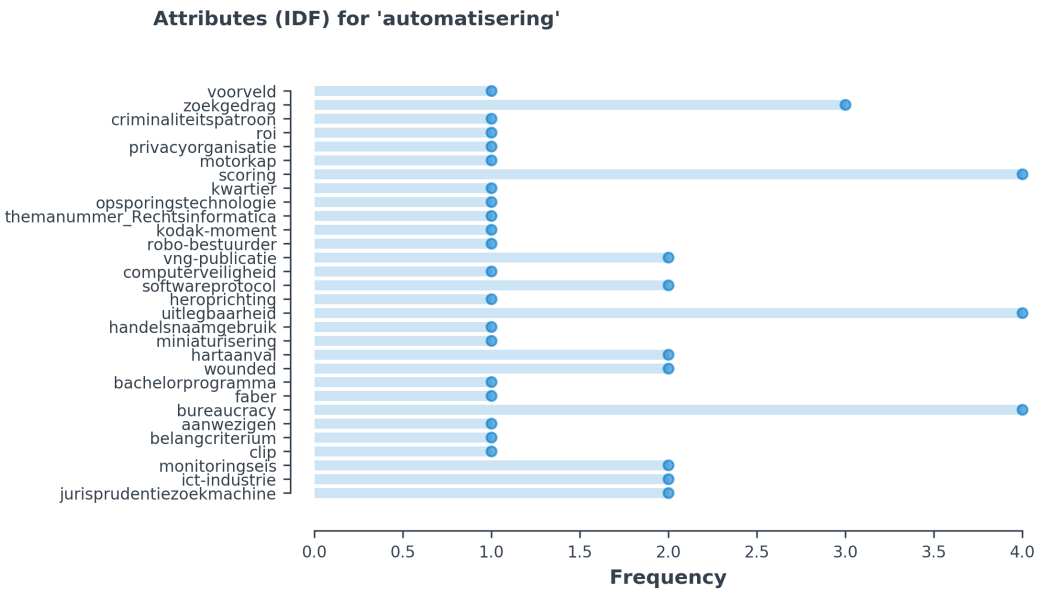
Some of the TF-attributes that are replaced are clearly relevant, such as "onderzoek", "ontwikkeling" and "studie", but they indeed seem quite general and it is thus no shock that their IDF scores are low. An interesting frequent attribute is "data". Query by navigation shows that this term comes from "Data Juridica", which is the name of a Wolters Kluwer database. Since this name then appears in many other documents, either part of or referring to this database, it is to be expected that it loses relevance according to the TF*IDF calculation. From the TF*IDF attributes, only the relevance of "bestuursvoorzitter" is not immediate, but other than that the selected attribute are of high quality. The meaning of two attributes were not immediately clear to me. "Effect" comes from "civiel effect", which is legal jargon for the prerequisites required to get access to further education to become for example a lawyer or judge. The other attribute is "po-punt", which refers to "opleidingspunten", i.e. educational points that each laywer has to earn to show his or her knowledge is up to date.

Interestingly enough, whereas the IDF-attributes of the previous query were of poor quality, there is now a selection of many relevant and interesting IDF attributes. Practically all attributes are directly connected with eduction. It is not clear why "deficiëntie" and "autopsie" show up, but apparently there are documents about a doctor's education as well (cf. the attribute "artsdiploma"). We see that using IDF can in some circumstances provide interesting attributes, but of course some terms have a high IDF not because they are interesting but because they contain a spelling mistake or have an unconventional spelling or abbreviation. "ongepubl", is an example, being an abbreviation for "ungepubliceerd".

5.2.3 Automatisering (n=130)





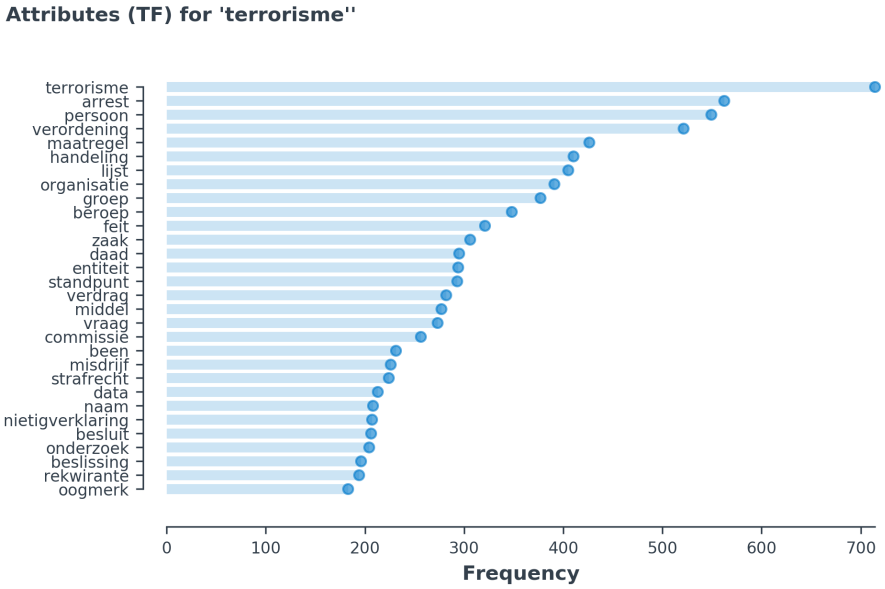


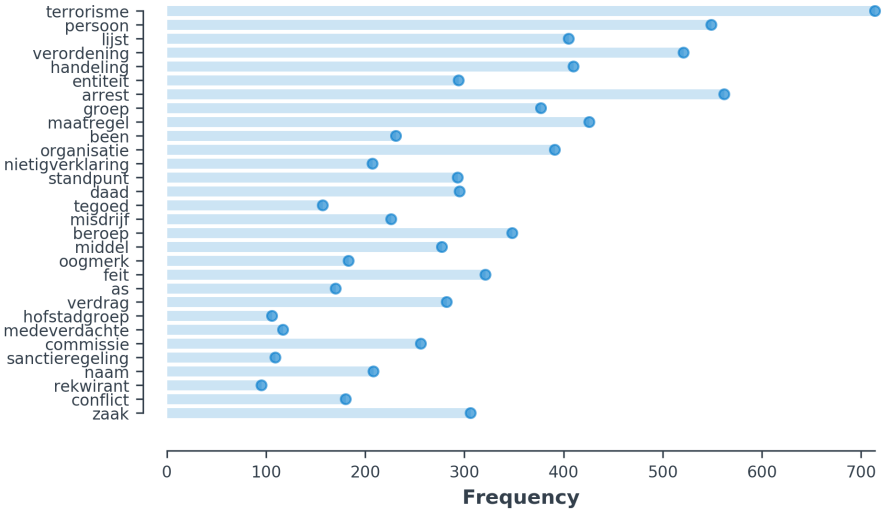
The TF-attributes contain several relevant keywords for the given query "automatisering", such as {data, digitalisering, automatisering, burger, computer, privacy, informatie, algorithme, bescherming }. There are also again a few formal terms that refer to legal practice itself, such as {vraag, zaak, preadvies, uitspraak, besluit}. Other terms are too general to be informative, such as {rechtspraak, zaak, belang, mens, mogelijkheid, rol, aantal, voorbeeld}.

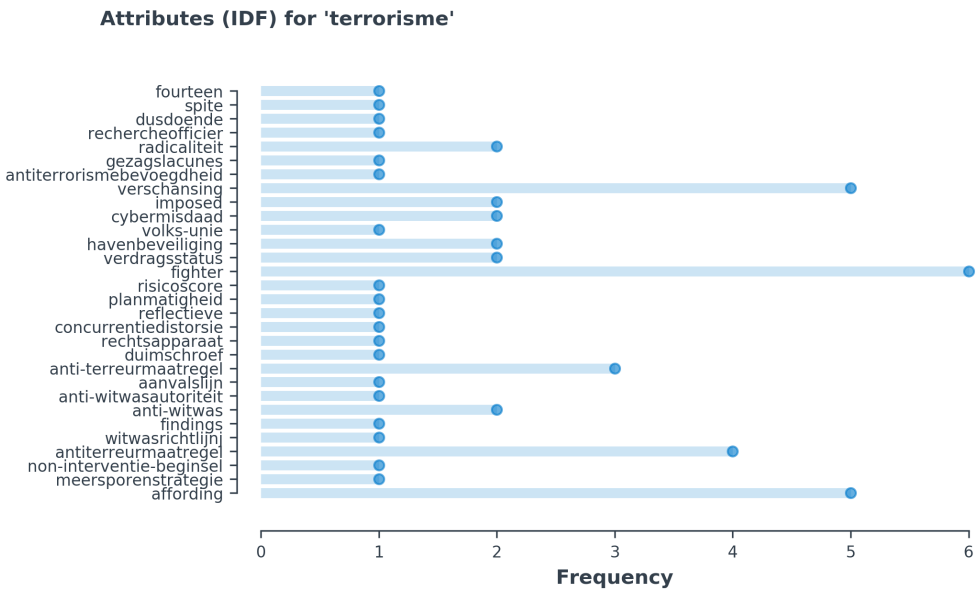
The TF-attributes {aantal, besluit, data, mogelijkheid, rol, uitspraak, voorbeeld} are replaced by the TF*IDF attributes {data-analyse, digitalis, homo, intelligentie, preadviseur, robot, technologie}. In general these substitutions make sense. For example, it is to be expected that a term such as "data" occurs often in texts about a "digital" topic. It is however not really descriptive and its disappearance when using TF*IDF is desirable. But regarding the substitutions, in particular the term "homo" is a surprising result, but a closer look shows that this comes from "homo digitalis". "Homo" thus does not refer to "homosexuality", which would be off-topic, but to the "digital human". "Digitalis" appears separately as an attribute as well, but since these terms clearly belong together this can be considerd a downside of not using longer n-grams. Another interesting term is "intelligentie". Texts matching this query concern the legal aspects of automatic decision making and concern the impact of artificial intelligence.

When we look at the IDF-attributes we see a few terms that are very interesting, such as {privacyorganisatie, opsporingstechnologie, computerveiligheid, uitlegbaarheid}. The relevance of the first three terms is immediately clear as they relate to digital security and privacy. The relevance of the term "uitlegbaarheid" (explainability) is less immediate, but nevertheless clear when we remember that the TF*IDF attributes learn us that many texts are about the impact of artificial intelligence. The term "explainability" is important for the legal aspects of artificial intelligence, because it relates to how decisions of intelligent digital systems can be explained and justified when they act as "black boxes". However, as for the previous queries, these few interesting terms each only occur in a single document, and the other attributes are too specific and quite off-topic.

5.2.4 Terrorisme (n=247)







Looking at the TF-attributes, we see a few relevant terms, such as {terrorisme, lijst, organisatie, groep}. “Organisatie” (organisation) and ”groep” (group) both zoom in on terroristic groups rather than individuals. “Lijst” (list) is interesting because various lists of known and wanted terrorists are maintained. Even though some “meta-attributes” pertaining to legal practice itself could be beneficial for navigating through different types of legal documents, in this case they completely dominate the top-n TF-attributes. Consider the terms {arrest, verordening, maatregel, handeling, beroep, feit, zaak, entiteit, verdrag, vraag, commissie, strafrecht, nietigverklaring, besluit, onderzoek, beslissing, rekwirante}. Apart from those, a few very general terms are left over: {middel, misdrijf, data, naam, oogmerk}.

A careful reader will notice that there is one very strange attribute present, namely “been”. When we use query by navigation to inspect documents containing “been”, we see that this attribute is an artefact of the preprocessing, that is however not trivially avoidable. The “terrorisme” query returns four documents that contain very long citations of English legal texts concerning terrorism, which in itself is understandable as terrorism is an international issue. However, in many sentences the phrase “has been” occurs, which is an inflection from the ubiquitous “to be”. The preprocessing pipeline can deal with languages other than Dutch, but coincidence has it that “been” means ”leg” in Dutch, and that moreover this is a noun that is thus selected as a candidate lemma. On top of this coincidence, it is also another coincidence that this confusion arises for a word that is extremely frequent in English but infrequent in the Dutch corpus that the IDF scores where computed over. Therefore we see that “been” does not only occur in the TF-attributes, but also in the TF*IDF-attributes.

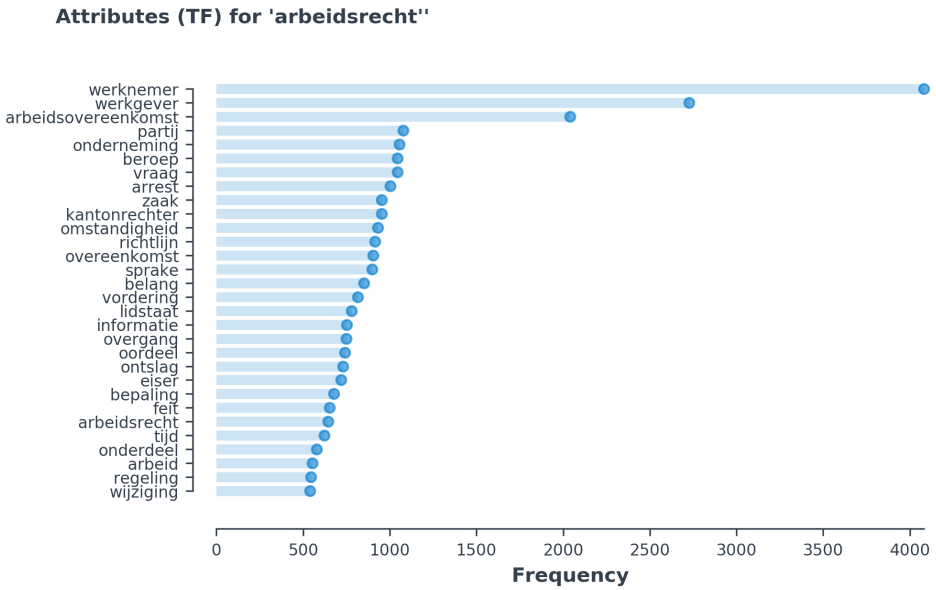
The TF-attributes {beslissing, besluit, data, onderzoek, rekwirante, strafrecht, vraag} are replaced by the TF*IDF-attributes {as, conflict, hofstadgroep, medeverdachte, rekwirant, sanctieregeling, tegoed}. These replacements are not a huge improvement, as most are still either general or formal, with the clear exception of “hofstadgroep”. The “Hofstad group” was an infamous Dutch terrorist network, so it makes sense that someone could navigate to a specific subset of documents concerning that topic. Unfortunately, the TF*IDF-attributes are still dominated by what I called “meta-attributes” that concern legal practice itself.

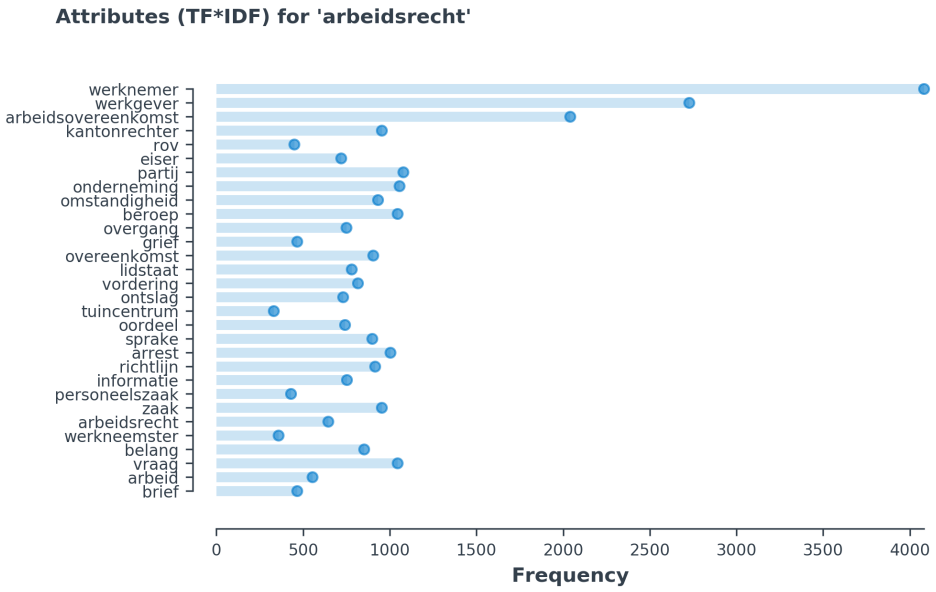
Another interesting observation is that in the TF*IDF-attributes we see the strange term “as” appear, which was absent in the TF-attributes. The underlying issue is the same as for the term “been”,

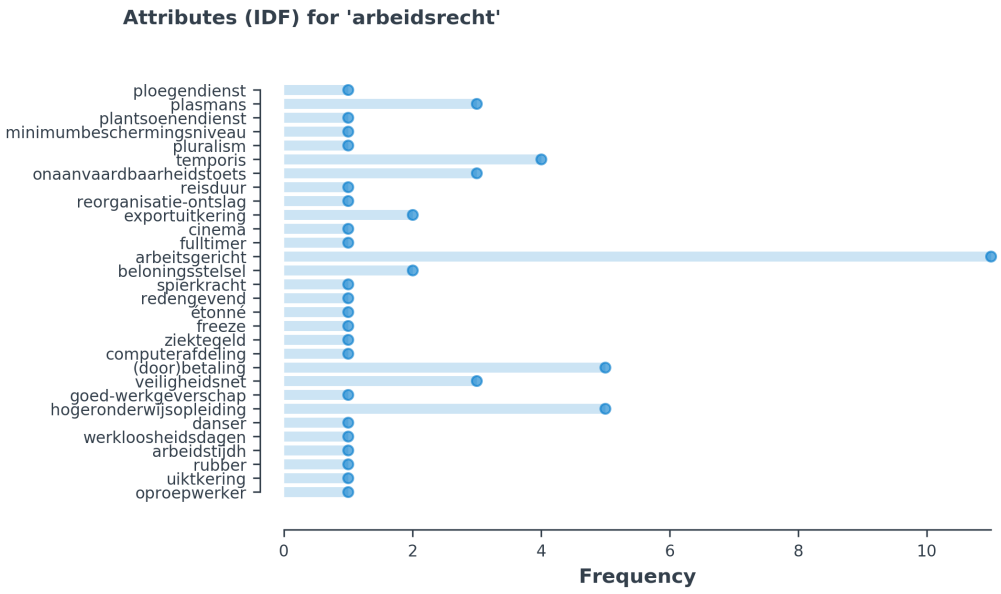
because “as” means “ash” in Dutch. The difference with the term “been” is that “as” is not as frequent in English as the word “been”, which is why it did not show up as an TF-attribute. However, it is not surprising that “ash” practically never occurs in the larger corpus of legal documents and thus has a high IDF score to boost it into the top-n TF*IDF-attributes.

When we look at the IDF-attributes, we see a similar pattern as with the previous queries. There are several quite interesting terms, such as {radicaliteit, gezagslacunes, antiterrorismebevoegdheid, cybermisdaad, duimschroef, anti-terreurmaatregel, antiterreurmaatregel, non-interventie-beginsel, meersporenstrategie}. You clearly see that IDF scores are very sensitive to alternative spellings (“anti-terreurmaatregel” vs “antiterreurmaatregel”). If a given word is spelled in an unconventional way or simply mispelled, its IDF value will be high. The same holds for English words that make it through the preprocessing pipeline, such as {fourteen, spite, fighter, findings, affording}. All attributes are very infrequent.

5.2.5 Arbeidsrecht (n=624)







A look at the TF-attributes shows that many attributes are quite general, even when they are relevant for the query "arbeidsrecht". Independent of the chosen relevance measure, this is not too surprising as "arbeidsrecht" is the least specific query of the examples discussed here, and returns way more documents than the other queries as a result. Relevant yet still quite general terms are {werknemer, werkgever, arbeidsovereenkomst, onderneming, beroep, zaak, overgang, ontslag, arbeid, regeling}. Again, there are quite many formal terms, namely {partij, kantonrechter, vraag, arrest, omstandigheid, richtlijn, overeenkomst, sprake, belang, vordering, lidstaat, oordeel, eiser, bepaling, feit, arbeidsrecht}. Other terms whose relevance is less clear, are {informatie, tijd, onderdeel, wijziging}.

The TF- and TF*IDF-attributes are quite similar. The formal and general TF-attributes {bepaling, feit, onderdeel, regeling, tijd, wijziging} are replaced by the TF*IDF-attributes {brief, grief, per-

soneelszaak, rov, tuincentrum, werkneemster}, which I would consider a slight improvement. The term "rov" is a juridical abbreviation for "rechtsoverweging", which refers together with a paragraph number to a legal consideration somewhere else in a relevant work. The appearance of the term "tuincentrum" (gardening center) is surprising. Query by navigation shows that there are two large articles describing cases about a gardening center going bankrupt, in which the term "tuincentrum" is very frequent. Being an uncommon word, a high IDF value combined with a surprisingly high frequency resulted in this word being marked as relevant for a set of 624 documents, even though it is only present in two very specific documents.

Some IDF-attributes are relevant and interesting, such as {ploegendienst, minimumbeschermingsniveau, reorganisatie-ontslag, exportuitkering, beloningsstelsel, ziektegeld, (door)betaling, veiligheidsnet, goedwerkgeverschap, werkloosheidsdagen, oproepwerken}. Again, we also see words from a different language (the German "arbeitsgericht") and typo's appear in this list (e.g. "uiktkering"). Some of the IDF-attributes occur in multiple documents, but most terms still only occur only once despite the set of documents being significantly larger.

5.2.6 Summary

In general TF*IDF clearly replaces some general TF-attributes with more specific and relevant attributes. In some queries relatively many attributes are formal juridical terms. In some cases this might be useful for lawyers, but ideally there is a balance between juridical terms and terms that are more descriptive of the content (rather than the type) of the documents. We see that some of these formal terms are filtered out when using TF*IDF, but not all of them. If this is not desired one could extend the list of domain-specific juridical stopping words. I would nominate a term such as "rov" as a domain specific stopword. But deciding which term counts as a stopword and which term could help make meaningful distinctions should be done in cooperation with legal experts and is a matter of fine-tuning.

5.3 Lattice properties

The purpose of selecting a limited amount of relevant attributes is done to ensure that conceptual navigation is meaningful and feasible. For conceptual navigation it is important that there are not so many concepts that it takes very long for users to specify a sufficiently narrowed down set of documents. By limiting the amount of formal context attributes, we restrict the total amount of concepts in the resulting concept lattice, but there is still variation in the amount of objects (i.e. documents) of a formal context. To estimate the feasibility of concept navigation we therefore need to get an insight about how the amount of concepts in a concept lattice grows with increasing document numbers, and how our attribute choice affects this increase rate. Ideally, if we have attributes that are a relevant representation of a particular domain, the amount of concepts hopefully grows somewhat proportional to the increase in documents.

There is expected to be a trade-off between how general or specific attributes should be. If all attributes are too general and apply to almost all texts, we can expect a combinatorial explosion of concepts when many of the $2^n - 1$ subsets apply to many documents. Having so many concepts and small variations between concepts makes conceptual navigation tedious. On the other hand, if our attributes are too specific, the concept lattice might not capture meaningful relations between documents, which could impact the quality and potential benefits of conceptual navigation.

Again, the quality of a concept lattice and by extension of the navigation over that lattice would ideally be tested with a large enough testing groups of legal professionals. This is not feasible within this thesis, so therefore I instead propose a crude quantitative measure to try to capture the aforementioned

expected trade-off. We have five example queries that return different amounts of documents. To get an insight into how the structure of the concept lattice changes for increasing document numbers, we can record the number of concepts per document. Intuitively, if for a given amount of documents there are more concepts, it will take longer to use these concepts to navigate to a small enough subset of documents. In other words, the average amount of concepts per document is inversely related to the *concept navigation speed*, which we can thus express as the average amount of documents we browse per concept (document/concept). We can record this measure for all three strategies for selecting relevant attributes.

Note that this attempt does not take into account that some domains might be more complex than others, independent of the size of documents.

Table 6: **Seksuele geaardheid**

"Seksuele geaardheid" (n=25)	Concepts	Concepts/doc.	Speed (docs/concept)
TF	119	4.76	0.21
TF*IDF	85	3.40	0.29
IDF	21	0.84	1.19

Table 7: **Beroepsopleiding**

"Beroepsopleiding" (n=38)	Concepts	Concepts/document	Speed (docs/concept)
TF	185	4.87	0.21
TF*IDF	64	1.68	0.59
IDF	21	0.55	1.81

Table 8: **Automatisering**

"Automatisering" (n=130)	Concepts	Concepts/document	Speed (docs/concept)
TF	3555	27.35	0.04
TF*IDF	1424	10.95	0.09
IDF	21	0.16	6.19

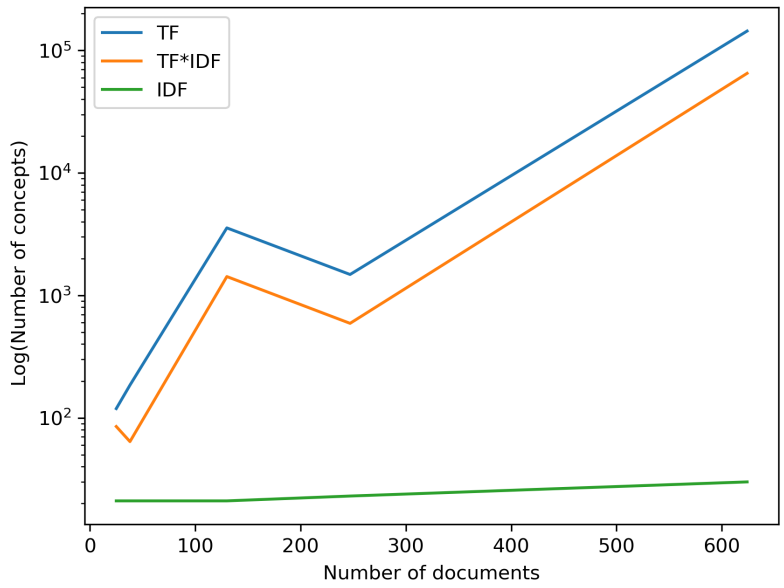
Table 9: **Terrorisme**

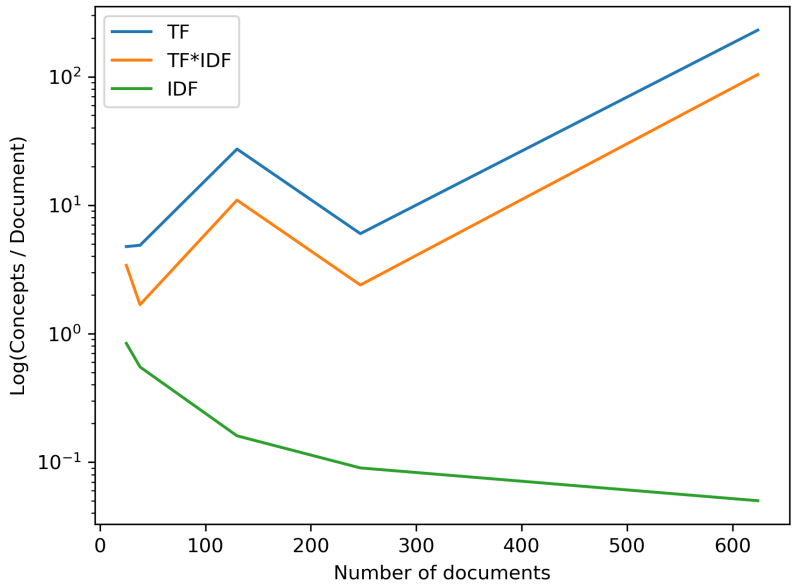
"Terrorisme" (n=247)	Concepts	Concepts/document	Speed (docs/concept)
TF	1481	6.00	0.17
TF*IDF	591	2.39	0.42
IDF	23	0.09	10.74

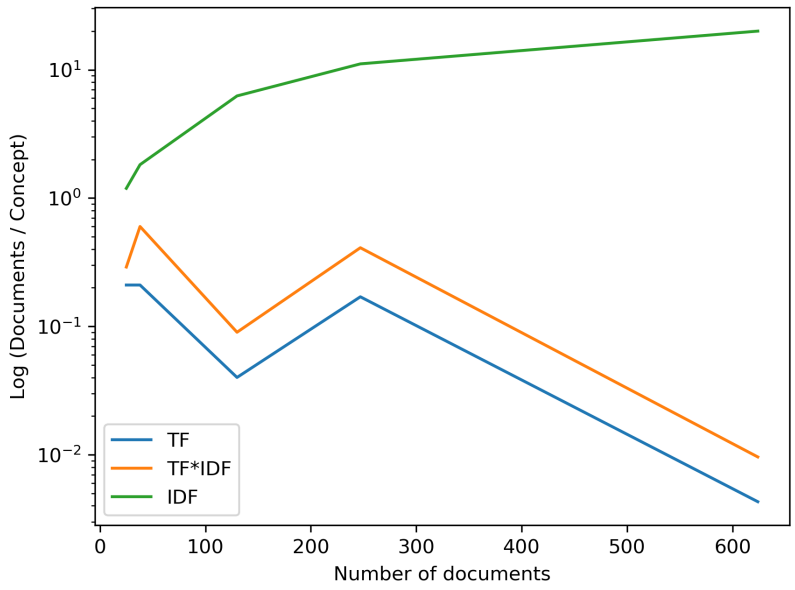
Table 10: **Arbeidsrecht**

"Arbeidsrecht" (n=624)	Concepts	Concepts/document	Speed (docs/concept)
TF	143906	230,62	0.0043
TF*IDF	64995	104	0.0096
IDF	30	0.05	21

5.3.1 Summary







Note that all these plots use a logarithmic scale on the y-axis and a linear scale on the x-axis. The logarithmic scale is used due to large differences in values.

We see that the absolute number of concepts grows exponentially with the number of documents, both for TF and TF*IDF, but that the amount of concepts grows faster for the TF-attributes. The amount of concepts for IDF attributes are however near constant, increasing from 21 concepts at 25 documents to 30 concepts for 624 documents. We see that for the query "terrorisme" there are actually way less concepts than for the "automatisering" query, despite having almost the double amount of documents. This shows us that the size of the concept lattice cannot be straightforwardly predicted based on the amount of documents. When interpreting results like these, we should also take into account how attribute counts are distributed over documents. For example, a term occurring twenty times in a set of documents could occur once in twenty documents, but could also occur twenty times in the same document. The former case will result in more concepts in the lattice than for the latter case. In the latter case that term, even though it might have a very high TF*IDF score, will only appear in one concept (excluding the infimum). The concept lattice of "terrorisme" is thus likely smaller because attributes on average occur in less documents, but then quite frequent in those documents. This could in turn be correlated with articles about specific topics being very lengthy.

When we look at the curve expressing the amount of concepts per document we see the same trends as in the previous plot, but this plot expresses a bit nicer how the amount of concepts for IDF-attributes decreases when compared to the increase in document size. That is, the amount of concepts for IDF-attributes is barely affected by an increase in documents and the complexity of the concept lattice is thus near constant. Notice that the values for IDF-attributes are all lower than 1. This means that in the average case a document does not occur at all in a meaningful concept in the concept lattice (i.e. outside of the supremum). The IDF-attributes are so specific that they only occurr in one to eight documents irregardless of how many documents we are considering. This makes IDF-attributes useless for concept navigation: each chosen concept will almost always immediately be the upper neighbour of the infimum, so the user can on average only make a single navigational choice before the navigation process terminates.

This is also represented in the (inverse) "navigational speed" values. A speed higher than 1 is too fast, because you exclude almost the whole set of documents with a single navigational choice. Lower

speed values mean that on average a concept choice rules out less documents. This is a very crude measure though that does not take into account the actual structure of the concept lattice.

5.4 Query by navigation

We have discussed the influence of relevancy measures and the number of documents on the size of concept lattices. We have also tried to interpret what the consequences on the structure of the concept lattice are likely to be. In this section we look at how the user sees these concept lattices in the "query by navigation" application written for this thesis. We try to interpret this against the idea of "navigational speed" from the previous section, but now taking into account considerations about the structure of the concept lattice.

In the previous section we considered that, against the general trend, the navigational speed of the "terrorisme"-lattice was higher than that of the "automatisering"-lattice, even though the former has almost double the number of documents. We hypothesized that this is because attribute occurrences for the "terrorisme"-lattice are divided over less documents than for the "automatisering"-lattice. Let's look at the most general concepts (so the lower neighbours of the supremum concept), and how many documents they describe.

```
Command >query
Start query with supremum

[R]== set()
├── 0. automatisering (52 documents)
├── 1. ontwikkeling (42 documents)
├── 2. vraag (42 documents)
├── 3. belang (36 documents)
├── 4. digitalisering (34 documents)
├── 5. gebruik (32 documents)
├── 6. mens (30 documents)
├── 7. bescherming (30 documents)
├── 8. rechtspraak (30 documents)
├── 9. systeem (29 documents)
├── 10. informatie (29 documents)
├── 11. gegeven (27 documents)
├── 12. computer (26 documents)
├── 13. burger (25 documents)
├── 14. privacy (25 documents)
├── 15. onderzoek (23 documents)
├── 16. zaak (23 documents)
├── 17. persoonsgegeven (22 documents)
├── 18. technologie (22 documents)
├── 19. overheid (21 documents)
├── 20. organisatie (20 documents)
├── 21. intelligentie (16 documents)
├── 22. algoritme (12 documents)
├── 23. robot (8 documents)
└── 24. besluitvorming (7 documents)
```

(a) Top level concepts for "automatisering" (TF*IDF attributes)

```
Command >query
Start query with supremum

[R]== set()
├── 0. terrorisme (193 documents)
├── 1. maatregel (60 documents)
├── 2. organisatie (53 documents)
├── 3. zaak (50 documents)
├── 4. persoon (43 documents)
├── 5. feit (37 documents)
├── 6. beroep (31 documents)
├── 7. verdrag (31 documents)
├── 8. arrest (28 documents)
├── 9. naam (25 documents)
├── 10. groep (24 documents)
├── 11. misdrijf (23 documents)
├── 12. lijst (23 documents)
├── 13. commissie (21 documents)
├── 14. middel (20 documents)
├── 15. verordening (17 documents)
├── 16. handeling (16 documents)
├── 17. daad (14 documents)
├── 18. tegoed (13 documents)
├── 19. entiteit (11 documents)
├── 20. nietigverklaring (11 documents)
├── 21. conflict (10 documents)
├── 22. hofstadgroep (6 documents)
└── 23. as (6 documents)
```

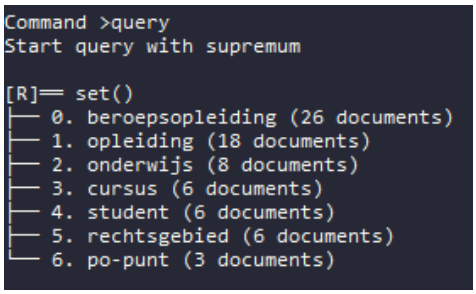
(b) Top level concepts for "terrorisme" (TF*IDF attributes)

The "automatisering" query returned 130 documents. Interestingly enough, if we look at the concept with the intent "automatisering" we see that only 52/130 documents contain the lemma "automatisering" itself. If we choose option 0 ("automatisering") in the navigation interface, we get a new set of conceptual choices. The concepts returning the most documents are "computer", which returns 13 documents of a total of 26 matching that term (see the top concepts in 3a), and "rechtspraak" (9/19) and "bescherming" (9/12). Interestingly, if we instead initially choose option 1 ("ontwikkeling"), the

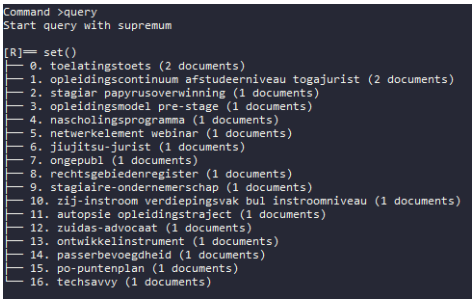
follow-up concepts correspond to the lemmas “vraag” (22/42), “belang” (22/36) and “digitalisering” (21/34). The important insight here is that these two initial choices that encompass the most documents give entirely different follow-up concepts, meaning that they immediately diverge and branch out.

This is not the case for the “terrorisme” query. The option returning the most documents is “terrorisme” (193/247). If we choose the option “terrorisme”, the top three follow-ups in document size are “maatregel”(47/60), “organisatie” (40/53) and “zaak” (38/50). Because the documents counts are still close to the total document matches of those terms in isolation (c.f. 3b), we can already infer that “terrorisme”, “maatregel”, “organisatie” and “zaak” occur together very often. Moreover, they are all part of the top four attributes ranked on the amount of documents they match, specifying the bulk of all results. In only two steps the amount of documents can be reduced from 247 to 38-47 documents, and without the need to “merge in” larger side branches later.

In other words, it seems that the “terrorisme”-concept lattice has less concepts per document because it has a more desirable structure, with a main strong branch that describes and distinguishes the bulk of documents quickly, whereas the “automatisering”-lattice starts with multiple equally large branches that all interact and need to “merge” into each other lower in the concept hierarchy. This finding is consistent with the hypothesis that the “terrorism”-lattice has a relatively higher “navigation speed” because terms co-occur together more often rather than being more or less uniformly spread out over the set of documents.



(a) Top level concepts for “beroepsopleiding” (TF*IDF attributes)



(b) Top level concepts for “beroepsopleiding” (IDF attributes), c.f. figure 5

It is also interesting to regard the difference in conceptual structure when using TF*IDF-attributes (a sane default choice) and IDF-attributes. We have seen that attributes co-occurring together relatively often tend to give a nicer concept structure that moves from general concepts to specific concepts without branching out too much, i.e. without becoming very wide. Indeed, for the TF*IDF variant, the user is presented with a nicely limited set of initial navigation choices (6 top level concepts of a potential 30 initial concept choices). What is also nice is that the user can either choose to enter a main path (option 0 and 1), or directly jump to a narrow set of documents (e.g. by choosing “rechtsgebied” or “po-punt”).

What we instead see for the navigation options when using IDF-attributes is that the concept lattice has an extremely wide and flat structure. It is wide because documents almost never share the IDF-attributes. It is extremely shallow because the attributes are so specific that they most of the time only occur in a single document. Using IDF-attributes there is thus not much to navigate, really. Option 1 is interesting because it displays the desirable property of the navigation process that multiple attributes are chosen at once when they all only occur in one particular set of documents (2 documents in this case), i.e. they always occur together.

Because the concept lattice is so small when using IDF-attributes, it is possible to draw a graphical visualization of this concept lattice, which nicely shows this extremely flat lattice structure.

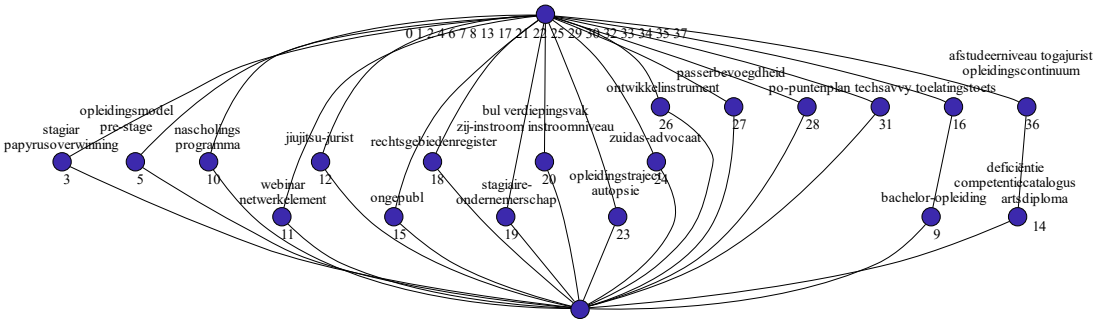


Figure 5: Visualized concept lattice of "beroepsopleiding" with IDF-attributes. Cf. 4b

6 Discussion

A main strategy for reducing concept lattice complexity of the proposed pipeline was to limit the amount of formal context attributes, but as a consequence the attention of this research shifted towards finding the most relevant attributes. If we want to capture meaningful structure in our concept lattices, we need to make sure the top-n selected attributes are not flooded by too common terms and stopwords. We have seen that compared to using TF, TF*IDF does a better job reducing the amount of common uninteresting attributes, and as a result TF*IDF scales slightly better with increasing document numbers. The occurence of general terms as such is not undesirable, because the beauty of query by navigation is that the user can start navigation using general and common terms, but then narrow down the query using more specific attributes. This works well as long as not all attributes are very general.

However, we have also seen that when queries return > 200 documents, the top-n attributes are nevertheless increasingly dominated by general terms and formal terms pertaining to legal practice itself in particular. A solution for the latter problem is to extend the list of domain specific stopwords in cooperation with legal professionals. But one might also wonder if there really is a problem with the current pipeline if this issue only becomes poignant when FCA is done on > 200 documents. Since the purpose of query by navigation is to support finding information in search results, we could simply make the assumption that this process is not started on more than 200 search results. Which user really cares about search result 201? Of course, if the list of search results is truncated, quantitative measures such as accuracy and recall do become very relevant as information is lost.

The alternative solution is to increase the amount of formal context attributes until the attributes that make it into the top-n attributes are specific enough. However, there are two reasons why this option was avoided in this thesis.

- Choosing n attributes puts a clear restriction on FCA complexity. Especially because the size of objects is variable by definition, it is beneficial to keep the number of attributes bounded.
- In the query by navigation application we present the user with an amount of choices whose upper bound is the amount of formal context attributes. If we for example increase the amount of attributes from 30 to 60, the user is potentially presented with 60 initial choices, which I would consider an information overload and as being detrimental to the usability of the navigation process.

Of course, when FCA is applied with other applications in mind, it might very well be appropriate to increase the number of attributes.

In the following section I discuss limitations of this research and suggestions for further research.

6.1 Limitations and suggestions for further research

N-grams A clear limitation of the current pipeline is that attributes are defined as keywords (1-grams). By limiting ourselves to keywords, we have effectively set up a baseline with regards to the quality of the concept lattice and the complexity of the overall pipeline. This baseline can serve as a point of comparison for further experimentation, for example with n-grams. My hypothesis is that using n-grams would be appropriate for the investigated legal texts, because some legal terms are meaningful only in their direct context. For example, a relevant attribute for the "beroepsopleiding" was "effect". By itself it is not immediately clear how the term "effect" is a relevant attribute for all documents returned by the "beroepsopleiding" query, but if we apply query by navigation to find the documents containing the effect-attribute, then we see this term comes from the phrase "civiel effect". This phrase is legal jargon for the prerequisites any education to become a legal professional (lawyer, judge etc.) must have. Suddenly, the connection to the search query "beroepsopleiding" (professional education) is completely obvious. Another example was for the query "automatisering", where in the TF*IDF attributes "homo" appeared. On further inspection, it turned out this keyword was part of "homo digitalis", which also explained the appearance of the attribute "digitalis", which by itself also does not make much sense.

However, I think there is also an argument for not considering too long n-grams. Imagine a search that returns a group of texts about digital privacy, including specific privacy issues about the online identities of homosexual people. It is not unlikely that "homo" would then be selected as an attribute in our formal context. But for the topic of the documents under consideration, it could very well be the case that in one half of the texts the term "homo" is used in the sense of "homo digitalis", and in another part of the texts in terms of homosexuality. The power of query by navigation would however be that you could disambiguate this query simply by choosing a follow up attribute. A user could choose "digitalis" to select texts about the "digital human" (as we've seen that it appears as an attribute as well), and a choice for another attribute could return documents specifically about the digital life of homosexual people. In this sense it is not a fundamental problem that "homo" appears alone, because FCA nicely treats this as part of a superconcept of the subconcepts containing different disambiguated meanings of the word "homo".

Experimenting with IDF Even though the IDF relevancy measure completely failed to capture common attributes in groups of documents, it nevertheless showed very interesting features that were often highly tailored to the search query. In other words, using IDF only does not return attributes that are *relevant to a subdomain* but instead features that are *interesting in some individual document.* Because of this extreme specificity, using only IDF results in extremely shallow concept lattices that do not allow for any sensible navigation. However, what could be interesting is experimenting with schemes where IDF scores have a larger influence in the TF*IDF calculation, where the goal is to find a balance between relevancy for the domain and uniqueness that results in interesting domain attributes. This could be appropriate for the legal domain this thesis worked on, because this might pick up jargon that is interesting from a legal perspective.

Using fuzzy approaches to deal with large formal contexts This thesis limited the amount of documents by only performing FCA after a search operation. The advantage of this approach is that all desirable properties of FCA are maintained, without the required computations becoming

completely unfeasible. If however in a follow-up application formal contexts nevertheless become too big, more fuzzy approaches that simplify the formal context can be explored. A good option are clustering techniques, which "can be applied to sets of objects, attributes or formal concepts" (Dias and Vieira, 2015, 7088). Cheung and Vogel for example apply Singular Value Decomposition (SVD) to term-document matrices in order to cluster together documents based on cosine similarity, thereby reducing the amount of objects for a formal context (Cheung and Vogel, 2005). Another option which is current state of the art in FCA research is to apply *interestingness measures* to formal concepts in order to select only the most interesting concepts. An overview of various interestingness measures specifically tailored to formal concepts can be found in Kuznetsov and Makhalova (2018).

It would also be interesting to compare FCA with another common unsupervised technique for extracting concept taxonomies, such as the quite similar technique of hierarchical clustering (e.g. de Knijff et al., 2013; Martino and Cantiello, 2009). Although there is plenty of literature on both FCA and hierarchical clustering separately, "there is actually a lack of comparative work concerning the task of automatically learning concept hierarchies with clustering techniques" (Cimiano et al., 2005, p.306). Such a comparison however would be fruitful for investigating what approach is more suitable for particular knowledge domains, such as law (also see Drymonas et al., 2010). A benefit of clustering techniques is that they abstract away from details and noise, but at the cost of losing the well-defined properties of formal concepts. Further research would have to point out how bad that is for information retrieval purposes.

Varying the amount of attributes Further experimentation can be done in varying the amount of formal context attributes used. Especially if a relevance measure is chosen that returns very specific terms (of which IDF is an extreme), it is possible to radically increase the amount of attributes, because we have observed that when using very specific attributes the size of the concept lattice barely increases for larger groups of documents. Moreover, increasing the number of (very specific) attributes might remedy the undesirable concept lattice flatness to some degree, as the chance becomes higher that attributes are present in multiple documents when you select more attributes.

Create a "golden standard" In order to complement the evaluation of concept lattices, a "golden standard" is required for (parts of) the Wolters Kluwer database. Benchmarking with respect to a golden standard is de facto the way to evaluate ontologies, but unfortunately such a golden standard is not available for Dutch legal texts. For this pipeline, such a golden standard would be an ontology handcrafted by legal experts, for several subdomains from the total corpus. Within the scope of this project the construction of such an ontology was not feasible, but in the case of follow-up research the construction of a golden standard would be desirable.

Incorporating feedback from user navigation choices During query by navigation users continually indicate what they consider to be relevant by choosing a particular concept over others for further exploration. This data could be used by Wolters Kluwer to infer user preferences and serve individualized search results.

7 Conclusion

To conclude, let us revisit the research questions of this project.

Firstly, we asked how Formal Concept Analysis can be used meaningfully with a large corpus of 0.5 million documents. After exploring various options, we chose the pragmatic solution to use an initial search to reduce the amount of documents FCA will be applied to. A main benefit of this "pipeline"

approach is that we can retain the exactness of FCA, whereas most other methods first build a formal context and then simplify or abstract away from it. The experiments done in this thesis suggest that we can plausibly assume that most queries reduce the document space to a size that can be handled by FCA quite well, and that is in particular still usable for query by navigation. Another advantage is flexibility: another company can use its own search system and database, without having to adjust the rest of the pipeline containing FCA and query by navigation.

Secondly, in addition to this strategy to limit the amount of objects in the formal context, we further reduce complexity by limiting the amount of attributes to only the most relevant ones. Therefore we explored how we can choose relevant formal context attributes that adequately represent the content of a set of documents. We explored three different relevance measures (TF, TF*IDF, and IDF) and manually evaluated the resulting attributes on relevance. The results are conform to expectations: TF-attributes are usually too general, whereas IDF-attributes are too specific, and TF*IDF-attributes strike a good balance between generality and specificity. We however also observed that especially when larger document sets are considered, the top-n TF*IDF-attributes are still increasingly dominated by formal legal terms. Some of those terms can effectively structure a set of documents and be useful in query by navigation, others might not. Further discussion with legal professionals could point that out and lead to the decision to put some terms on a domain-specific list of stop words that will not be considered as attributes.

Thirdly, we discussed how different choices for formal context attributes might affect concept navigation by the user. In particular, we considered how properties of concept lattices change as a result of which type of attributes are chosen for the formal context, and what properties are desirable for concept navigation. A property that is important for navigation is the structure of a concept lattice. A structure that is flat and wide will present the user with too many initial choices, and with very little options for further navigation. Conversely, a structure that is deep and narrow will require too many navigational choices to narrow down the query. A concept lattice structure that suits query by navigation ideally balances these two extremes.

However, the structure of a concept lattice can not simply be explained based on the amounts of concepts, but depends on multiple factors, including the type and amount of chosen attributes, but also the distribution of attribute presence over the corpus. Neither can non-trivial concept lattices be visualized in a way that would support interpretation. Despite the unavailability of quantitative measures we nevertheless provided indications on how to evaluate which of the resulting concept lattices is better for query by navigation. As a crude measure that does not take lattice structure into account, we indicated a "conceptual navigation speed". This showed that when using IDF-attributes most documents do not appear in a meaningful place in the concept lattice, and that query by navigation is expected to take longer when using TF-attributes than for TF*IDF-attributes. We then manually evaluated exemplary lattices based on the distribution of documents over concepts, as reported by the query by navigation application. This provided further support for a relevance measure such as TF*IDF that balances generality with specificity, since an ideal browsable structure seems to contain few very general concepts that appear in almost all documents, and then incrementally more specific attributes.

8 Bibliography

References

Ch. Aswani Kumar and S. Srinivas. Concept lattice reduction using fuzzy K-Means clustering. *Expert Systems with Applications*, 37(3):2696–2704, 2010. ISSN 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2009.09.026. URL http://www.sciencedirect.com/science/article/pii/S0957417409008070.

A. Buzmakov, S. Kuznetsov, and A. Napoli. Sofia: how to make FCA polynomial? In *Proceedings of the 4th International Conference on What can FCA do for Artificial Intelligence?-Volume 1430*, pages 27–34. CEUR-WS. org, 2015.

K.S.K. Cheung and D. Vogel. Complexity Reduction in Lattice-Based Information Retrieval. *Information Retrieval*, 8(2):285–299, 2005. DOI: 10.1007/s10791-005-5663-y.

P. Cimiano, S. Staab, and J. Tane. Automatic acquisition of taxonomies from text: FCA meets NLP. In *Proceedings of the International Workshop & Tutorial on Adaptive Text Extraction and Mining held in conjunction with the 14th European Conference on Machine Learning and the 7th European Conference on Principles and Practice of Knowledge Discovery in Data*, 2003.

P. Cimiano, A. Hotho, and S. Staab. Learning Concept Hierarchies from Text Corpora using Formal Concept Analysis. *Journal of Artificial Intelligence Research TA - TT -*, 24:305–339, 2005. DOI: 10.1613/jair.1648LK-https://ru.on.worldcat.org/oclc/7782984126.

V. Codocedo, C. Taramasco, and H. Astudillo. Cheating to achieve formal concept analysis over a large formal context. In *The Eighth International Conference on Concept Lattices and their Applications-CLA 2011*, pages 349–362, 2011.

J. G. Conrad and L. K. Branting. Introduction to the special issue on legal text analytics. *Artificial Intelligence and Law*, 26(2):99–102, 2018. DOI: 10.1007/s10506-018-9227-zLK-https://ru.on.worldcat.org/oclc/7897840039.

J. de Knijff, F. Frasincar, and F. Hogenboom. Domain taxonomy learning from text: The subsumption method versus hierarchical clustering. *Data & Knowledge Engineering*, 83:54–69, 2013. ISSN 0169-023X. DOI: https://doi.org/10.1016/j.datak.2012.10.002. URL http://www.sciencedirect.com/science/article/pii/S0169023X12000973.

S. M. Dias and N. J. Vieira. Concept lattices reduction: Definition, analysis and classification. *Expert Systems with Applications*, 42(20):7084–7097, 2015. ISSN 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2015.04.044. URL http://www.sciencedirect.com/science/article/pii/S0957417415002869.

S. M. Dias and N. J. Vieira. A methodology for analysis of concept lattice reduction. *Information Sciences*, 396:202–217, 2017. ISSN 0020-0255. DOI: https://doi.org/10.1016/j.ins.2017.02.037. URL http://www.sciencedirect.com/science/article/pii/S0020025517305388.

E. Drymonas, K. Zervanou, and E. G. M. Petrakis. Unsupervised ontology acquisition from plain texts: the OntoGain system. In *International Conference on Application of Natural Language to Information Systems*, pages 277–287. Springer, 2010.

F. Eynde. Part of Speech Tagging en Lemmatisering van het Corpus Gesproken Nederlands. Technical report, Centrum voor Computerlinguïstiek, K.U. Leuven., jan 2004.

F. A. Grootjen. *A Pragmatic Approach to the Conceptualisation of Language*. PhD thesis, Radboud University, 2005.

D. I. Ignatov. Introduction to formal concept analysis and its applications in information retrieval and related fields. In *Communications in Computer and Information Science*, volume 505, pages 42–141. Springer Verlag, 2015. ISBN 9783319254845. DOI: 10.1007/978-3-319-25485-2_3.

M. G. Ilieva and O. Ormandjieva. Natural language processing and formal concept analysis technologies for automatic building of domain model. In *Proceedings of the 11th IASTED International Conference on Software Engineering and Applications*, pages 445–452. ACTA Press, Cambridge, Massachusetts, 2007. ISBN 978-0-88986-706-2. URL http://dl.acm.org/citation.cfm?id=1647636.1647713.

S.O. Kuznetsov and T.P. Makhalova. On interestingness measures of formal concepts. *Information Sciences*, abs/1611.0(442):202–219, 2018. URL http://arxiv.org/abs/1611.02646.

C. Lindig. Fast Concept Analysis. In *Working with Conceptual Structures - Contributions to ICCS 2000*, pages 152–161. Shaker Verlag, feb 2000.

B. Martino and P. Cantiello. Automatic ontology extraction with text clustering. In *Intelligent Distributed Computing III*. 2009. ISBN 3642032133.

J. Medina. Relating attribute reduction in formal, object-oriented and property-oriented concept lattices. *Computers & Mathematics with Applications*, 64(6):1992–2002, 2012. ISSN 0898-1221. DOI: https://doi.org/10.1016/j.camwa.2012.03.087. URL http://www.sciencedirect.com/science/article/pii/S0898122112002921.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine Learning in {P}ython. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

K. Spark Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, 28(1):11–21, jan 1972. ISSN 0022-0418. DOI: 10.1108/eb026526. URL https://doi.org/10.1108/eb026526.

K. Sumangali and Ch. Aswani Kumar. Concept Lattice Simplification in Formal Concept Analysis Using Attribute Clustering. *Journal of Ambient Intelligence and Humanized Computing*, 2018. ISSN 1868-5137. DOI: 10.1007/s12652-018-0831-2LK-https://ru.on.worldcat.org/oclc/7626005760.

A. Van den Bosch, B. Busser, S. Canisius, and W. Daelemans. An efficient memory-based morphosyntactic tagger and parser for Dutch. In *Computational Linguistics in the Netherlands: Selected Papers from the Seventeenth CLIN Meeting*, pages 99–114. jan 2007.

K. Van der Sloot, A. Van den Bosch, and M. Van Gompel. Frog: An advanced Natural Language Processing suite for Dutch. URL https://languagemachines.github.io/frog/.

M. Van Gompel. LaMachine. URL https://proycon.github.io/LaMachine/.

R. Wille. Restructuring lattice theory: an approach based on hierarchies of concepts. In S. Ferré and S. Rudolph, editors, *International Conference on Formal Concept Analysis*, pages 314–339. Springer, 2009.

A. Wyner, R. Mochales-Palau, M.F. Moens, and D. Milward. Approaches to text mining arguments from legal cases. In *Semantic processing of legal texts*, pages 60–79. Springer, 2010.

9 Appendix: Notes on code implementation

A major portion of the effort that went into this project was time spent on implementating the whole pipeline from A to Z. Since this is not a coding project, I will not explain the code into detail, but I will provide an overview of the used classes and project dependencies.

Email me at my university email `edwin.wenink@student.ru.nl` to get access to the code.

9.1 Used modules

Python 3.6.7 was used. Python built-in modules are not mentioned.

```
beautifulsoup4 4.7.1 used for xml parsing
concepts 0.8 query by navigation is built on top of FCA functions provided by this package
nltk 3.4.1 not used heavily, can be easily replaced
python-frog 0.3.7 python wrapper for FROG
scikit-learn 0.20.3 used for TF*IDF and IDF
```

One method for getting all required dependencies in one go is to install the LaMachine software distribution (Van Gompel), which also contains all necessary dependencies for the python wrapper for FROG. `python-frog` can however also be installed independently. `concepts` is an implementation of the "Fast Concept Analysis" algorithm of Lindig (Lindig, 2000), and is used as the basis of query by navigation.

9.2 Classes

9.2.1 pipeline.py

The end user calls this script, which imports all required classes and other dependencies, and prompts the user to set options for the pipeline.

```
----------------------------------------

          QUERY BY NAVIGATION USING:

          ___       ___          ___
         /\__\     /\__\        /\  \
        /:/ _/_   /:/  /       /::\  \
       /:/ /\__\ /:/  /       /:/\:\  \
      /:/ /:/  //:/  /  ___  /:/ /::\  \
     /:/_/:/  //:/__/  /\__\/:/_/:/\:\__\
     \:\/:/  / \:\  \ /:/  /\:\/:/  \/__/
      \::/__/   \:\  /:/  /  \::/__/
       \:\  \    \:\/:/  /    \:\  \
        \:\__\    \::/  /      \:\__\
         \/__/     \/__/        \/__/

      BY EDWIN WENINK
              @Radboud University

----------------------------------------
```

```
The pipeline consists of:

- Optional: Preprocess whole corpus for IDF scores
- Optional: Selecting a subset of texts matching a keyword
- Optional: Preprocessing that subset
- Computing a formal context and concept lattice
- Query by navigation of that concept lattice

Indicate whether to perform the optional steps.
```

The user is guided through these steps and asked whether to perform the optional steps. For each step, it is explained what it does and why you might be able to skip it. Effort is put into avoiding work wherever possible. The pipeline writes intermediate results to standard locations, so that when a user chooses to not repeat a step, pre-computed results are loaded instead.

- The first step of computing the IDF scores over the whole corpus is very expensive, so you only want to do this once. Uses the `ExtractAll` class from `extract_all_text.py` (18 LOC), and the `ComputeIDF` class from the `compute_idf.py` script (70 LOC).
- The second step will search through the whole corpus (in this case 0.5 million documents) for all documents matching the keyword of interest, and write them to disk in a folder labeled with the ID and name of the keyword. Uses the `Selector` class from `preprocessing.py` (450 LOC).
- The third step will perform all preprocessing steps involving analysis of the documents in the folder from the previous step. Uses the `PreProcessor` class from `preprocessing.py`. End result are files with statistics, keywords, and TF- TF*IDF- and IDF-attributes.
- The fourth step calls the `FCA` class from `fca.py` (440 LOC). This computes a formal context and a concept lattice using either TF, TF*IDF, or IDF attributes. Also handles serialization and loading of concept lattices with the `LatticeSerialization` class from `serialize_lattice.py` (75 LOC).
- The fifth step calls the interactive `QueryByNavigation` class from `query_by_nav.py` (120 LOC).

That roughly summarizes the interaction between classes.

9.2.2 extract_all_text.py

Uses Frog to parse the whole corpus. The parsed lemmas are written to disk for later use. Because the preprocessing steps rely on Frog parsing, the IDF scores also need to be computed on Frog-parsed words.

9.2.3 compute_idf.py

Computes an IDF vector over all lemmas extracted from the corpus. A very important detail here is that a custom tokenizing regular expression is provided, because the scikit-learn default tokenizer does not correctly handle compound words. E.g. the word “advocaat-stagiar” is handled as two tokens by scikit-learn, which causes issues because Frog (correctly) considers this to be one word. This issue is correctly handled by this script, using the approach of Frog. The resulting word count and IDF-vector are serialized and written to disk for later use.

9.2.4 preprocessing.py

This script contains two classes, `Selector` for searching texts matching the query, and `PreProcessor` for analysing the contents of the retrieved raw xml files. The PreProcessor extracts relevant text sections from the xml files (title, search summary, body content) and applies several filtering actions to make sure that clean plain text is outputted and written to disk. From each text files noun lemmas are extracted using Frog, with the exclusion of lemmas defined as stop words. From the resulting set of lemmas three sets of attributes are computed, based on TF, TF*IDF and IDF calculations. For TF I used an nltk function to compute word frequencies. For TF*IDF and IDF computations I worked with efficient sparse matrices and used the vector models from the previous step.

Note that this whole script depends on the layout of the Wolters Kluwer database, and has to be adapted if the pipeline is used for different purposes.

9.2.5 fca.py

Contains functions for computing the formal context and concept lattices, and various helper functions that are needed for query by navigation, such as returning the lower neighbours of a concept. The user is presented with the choice whether to use the TF, TF*IDF, or IDF attributes. Because computation of a concept lattice can be very expensive, this script makes sure to always load the formal context and the corresponding concept lattice if they are already computed. This requires concept lattices to be serialized. However, big concept lattices cannot be serialized using the native python serialization module `pickle`, and unfortunately the `concepts` package also did not support this.

9.2.6 serialize_lattice.py

However, I contacted the author of the `concepts` package, Sebastian Bank from Leipzig University, to suggest this serialization feature and explain my issue and use case. He helped me and suggested a code sample to solve my problem. I tested the correctness of the code, and used it to serialize concept lattices to the json format. I am glad to say that this feature is now *merged into the master branch of the `concepts` package*. I am currently in the process of helping the author test the performance of serialization for json on the concept lattices I computed for this project.

9.2.7 query_by_nav.py

This script provides a command line REPL application to interactively traverse the computed concept lattice. The following print shows the available commands:

```
You can use the following commands:

- query [args]: start a new query with keywords [args], default=supremum
- [int]: expand query with option [int]
- show: show document names for current query
- inspect [id]: inspect contents of file with id [id]
- help: show this message
- quit: stop navigation
```

Available query options are displayed to the user in the form of a nice ascii tree, For each available option it is displayed how many documents are in the extent of the concept, if you would choose that option.