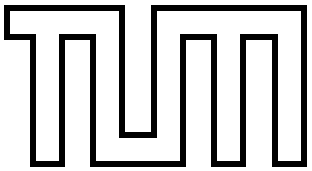


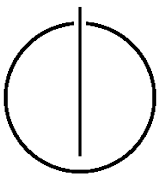
DEPARTMENT OF INFORMATICS

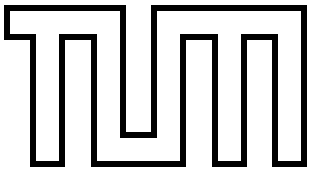
TECHNICAL UNIVERSITY OF MUNICH

Master's Thesis in Information Systems

Semantic Text Matching of Company Policies and Regulatory Documents using Text Similarity Measures

Christoph Erl





DEPARTMENT OF INFORMATICS

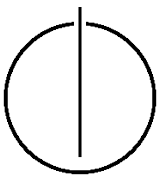
TECHNICAL UNIVERSITY OF MUNICH

Master's Thesis in Information Systems

Semantic Text Matching of Company Policies and Regulatory Documents using Text Similarity Measures

Semantic Text Matching von Unternehmensrichtlinien und Regulatorischen Dokumenten mittels Textähnlichkeitsmaßen

Author:	Christoph Erl
Supervisor:	Prof. Dr. Florian Matthes
Advisor:	Jörg Landthaler
Submission Date:	February 15, 2018



I confirm that this master's thesis is my own work and I have documented all sources and material used.

Munich, February 15, 2018 Christoph Erl

Acknowledgments

At this point, after an intensive period of six months, I would like to give many thanks to all the people who supported me during my thesis:

At first, I would like to offer my special thanks to my advisor Jörg Landthaler. He inspired me with constructive discussions, gave valuable feedback and guided me in the right direction if needed.

I would like to acknowledge my gratitude to Prof. Dr. Florian Matthes for his feedback and the opportunity to write my thesis at his chair Software Engineering for Business Information Systems.

My sincere thanks goes to the founders of Alyne who provided me the opportunity to join this exciting practice-oriented research project: Manuel Reil, Karl Viertel, Matthias Danner and Stefan Sulistyo. Thank you very much for the great time at Alyne, for participating in the expert interviews and for proofreading my thesis.

Finally, I must express my very profound gratitude to my parents, my brothers and my girlfriend for providing me with unfailing support and encouragement through the process of writing this thesis.

Christoph Erl

Abstract

Companies need to comply to international standards and regulations - both growing in number and complexity. Derived from these regulatory documents, large enterprises typically maintain individual company policies but do not keep their original references which are required, however, for future audits or updates. *Alyne GmbH* provides a solution with intermediate, so-called control statements that map controls from both regulatory documents and company policies. We develop a recommender system that aims to support regulatory experts at mapping controls with semantically similar control statements using text similarity approaches. This work investigates to which degree this recommender system can solve the mapping problem. Finally, we evaluate the results with datasets from real-world regulatory documents.

Zusammenfassung

Unternehmen müssen internationale Standards und Richtlinien einhalten, die jeweils stetig in ihrer Anzahl und Komplexität wachsen. Große Firmen pflegen in der Regel eigene, von diesen regulatorischen Dokumenten abgeleitete Unternehmensrichtlinien, halten aber keine Referenz zur Orginalquelle. Diese wird allerdings im Falle einer Prüfung oder Aktualisierung erforderlich. Die *Alyne GmbH* löst dieses Problem mit intermediären, sogenannten Control Statements, die Richtlinien von regulatorischen Dokumenten und Unternehmensrichtlinien miteinander verknüpfen. Ziel ist es, ein Recommender System zu entwickeln, das Experten dabei unterstützt, semantisch ähnliche Richtlinien mit Control Statements zu verlinken. Hierfür verwenden wir Textähnlichkeitsmaße. Diese Arbeit untersucht, inwieweit dieses Recommender System das erwähnte Verknüpfungsproblem löst. Schließlich werden die Ergebnisse mit realen Daten von regulatorischen Dokumenten evaluiert.

Contents

Acknowledgements iv

Abstract v

1. Introduction 1
- 1.1. Motivation . . . 2
  - 1.1.1. Explicit Mapping of Controls . . . 2
  - 1.1.2. Problem Statement . . . 3
  - 1.1.3. Proposed Solution . . . 4
- 1.2. Research Questions . . . 5
- 1.3. Research Approach . . . 6
  - 1.3.1. Iterative Approach . . . 6
  - 1.3.2. Ground Truth . . . 7
  - 1.3.3. Evaluation Criteria . . . 8
- 1.4. Structure of the work . . . 9

2. Semantic Text Matching 10
- 2.1. Definition . . . 10
- 2.2. Related Work . . . 11

3. Natural Language Processing 12
- 3.1. Preprocessing Technologies . . . 12
- 3.2. Text Similarity Approaches . . . 14
  - 3.2.1. Term Frequency - Inverse Document Frequency . . . 14
  - 3.2.2. Word2Vec (Word Embeddings) . . . 15
  - 3.2.3. Doc2Vec (Document Embeddings) . . . 17

4. Implementation Recommender System 19
- 4.1. Requirements . . . 19
- 4.2. System Structure . . . 20
  - 4.2.1. Configuration . . . 20
  - 4.2.2. Matcher . . . 22
  - 4.2.3. Data Sets . . . 23

5. Implementation GUI Citadel 24
- 5.1. Setup . . . 24
- 5.2. Features . . . 26
  - 5.2.1. Evaluation Management . . . 26
  - 5.2.2. Evaluation Analysis . . . 27

6. Evaluation **31**
6.1. Individual Evaluation 32
6.1.1. Preprocessing Analysis 32
6.1.2. Meta Information 35
6.1.3. Corpora Analysis 38
6.1.4. Context Information 40
6.2. Overall Evaluation 42
6.2.1. Text Similarity Approach Analysis 42
6.2.2. STM Problem Analysis 43

7. Conclusion and Future Work **49**

Bibliography **51**

List of Figures **52**

List of Tables **54**

Appendix **56**

A. Result Tables **56**

1. Introduction

Corruption, money laundering and fraud cause massive economic damage to companies year after year. And new forms of crime emerge through the progressive digitization such as computer fraud or espionage and interception of confidential company data.

Therefore, governments and professional associations develop standards and regulations in order to combat this economic crime and to reduce the legal, operational as well as financial risk for enterprises.

Deriving controls from regulatory documents and managing them in form of an internal controls system (company policies) is a legal requirement for the corporate governance of many companies in Germany for financial controls (BilMoG, Bilanzrechtsmodernisierungsgesetz) and also a regulatory requirement in regulated industries such as banking where this is defined in the KWG (Kreditwesengesetz) in Germany.

This derivation creates an implicit reference between a control of a company policy and the corresponding control of a regulatory document, as illustrated in Figure 1.1. The problem that arises now is that companies often do not store the explicit reference to the original control of a regulatory document which is required, however, for future audits or updates.

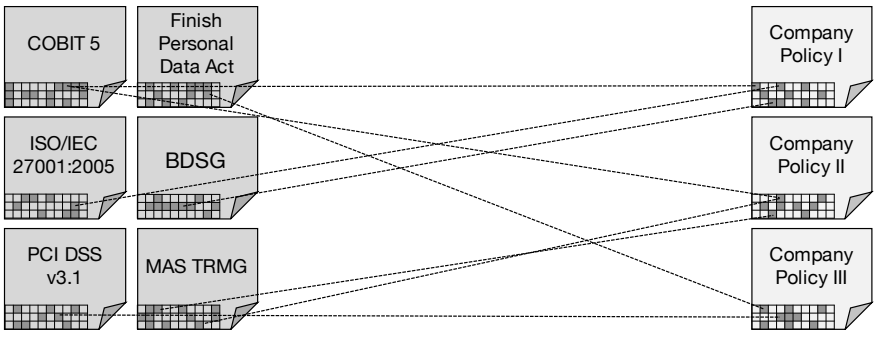


Figure 1.1.: Implicit References between Regulatory Documents and Company Policies

1.1. Motivation

The motivation of this thesis is driven by a concrete business problem that our industry partner *Alyne GmbH*[1] faces. *Alyne* is a young RegTech company that was launched in Munich in the year 2015. Their product is a B2B software as a service that supports organisations to manage their cyber security, risk management and compliance.

Our goal is to support regulatory experts of *Alyne* to create explicit references between semantically related controls from company policies and regulatory documents.

This section explains how *Alyne* solves the problem of explicit mappings, describes the business problem and proposes the solution idea.

1.1.1. Explicit Mapping of Controls

Enterprises typically extract and derive controls from regulatory documents and maintain a smaller number of individual company policies that are relevant for their business.

Table 1.1 shows two example controls in the field of password management, one from a regulatory document and one from a company policy:

Type	Reference	Control
Regulatory Document	PCI DSS 3.1 8.2.5	Do not allow an individual to submit a new password that is the same as any of the last four passwords he/she has used.
Company Policy	Example	The user password must be at least 8 characters long with at least one special character and must not same as any of his/her last three passwords.

Table 1.1.: Example controls from regulatory documents and company policies

This company policy control seems to be derived from the regulatory document control and additionally extended by password charactistics. The overlapping control - passwords must not be same as any of the previous passwords - only differs in the specified number of pervious passwords. This is an example for a potential implicit reference.

To make it explicit, *Alyne* manually extracted key statements from various regulatory documents of different topics (data privacy, fraud prevention etc.) and created a collection of (so far) 879 well defined *controls statements*. Controls statements store explicit references to related controls from both regulatory documents and company policies, as shown in Figure 1.2.

The corresponding control statement for the example above is “user passwords shall be prevented from being changed to any of the previous n passwords”.

[1] `https://www.alyne.com/en/`, last accessed February 2018

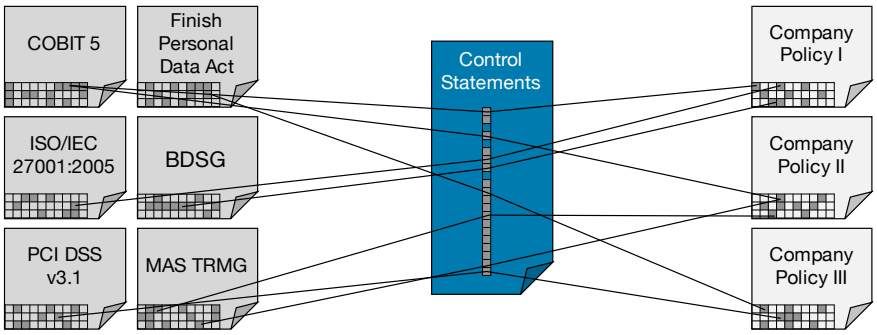


Figure 1.2.: Explicit References between Regulatory Documents and Company Policies

The generalization of controls from regulatory documents to distinct control statements counteract the today's growing number and complexity of regulations. *Alyne*'s customers can exclusively focus on them but the explicit references still allow to see to which documents they comply and to which degree.

1.1.2. Problem Statement

The collection of control statements and their explicit references is the unique selling proposition of *Alyne*. The extension and maintenance, however, is a very labor intensive process.

Let us consider a relevant selection of work routines of an *Alyne* expert in Figure 1.3.

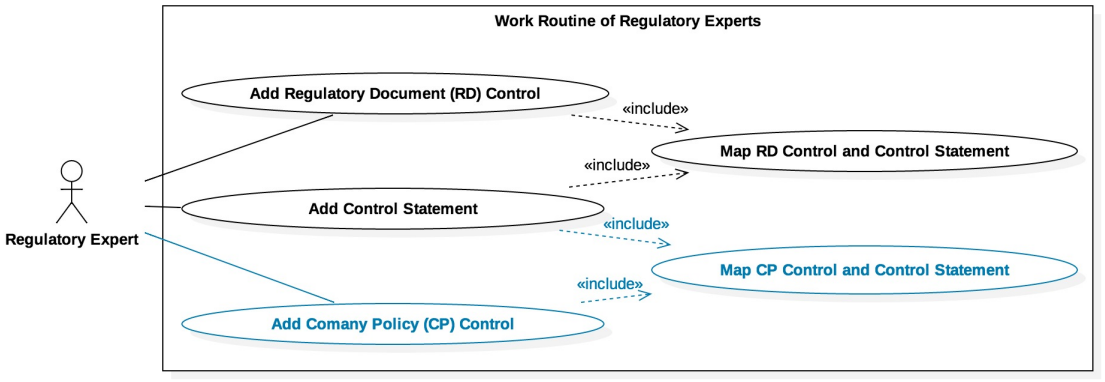


Figure 1.3.: Work Routine of Regulatory Experts

Regulatory experts need to add regulatory document controls, company policy controls as well as control statements. Since the number of regulatory documents and control statements is finite and comparatively small, it would be acceptable to maintain and map them by hand.

Company policies, however, are potentially infinite due to an unlimited number of customers. While the extraction of controls from policies can be achieved by simple algorithms (assuming controls are equal to text paragraphs), expert knowledge is required to reference a control to its corresponding control statement to ensure not to create wrong mappings and thus achieve a high data quality.

The problem statement that arises now is: How can we support *Alyne*'s regulatory experts in their work routine, in particular, to map company policy controls to control statements?

It is noted that, although the focus is on mapping company policies, throughout the work we do not differentiate between company policy and regulatory document controls because both are assumed to be equal.

1.1.3. Proposed Solution

To support *Alyne*'s regulatory experts in mapping company policies to control statements, the thesis **proposes a recommender system using text similarity approaches**.

Whenever an expert intends to map a control, the recommender system sorts the collection of 879 control statements according to their semantic similarities to the control. The intended effect is that experts

- do not need to check all control statements but only the top results (time savings) and
- find control statements which they might have forgotten (improved data quality).

A fully automatic approach instead of a recommender system was considered as not realisitic because the semantic similarity of controls is hard to capture and requires domain knowledge.

The principle of the recommender system is illustrated in Figure 1.4.

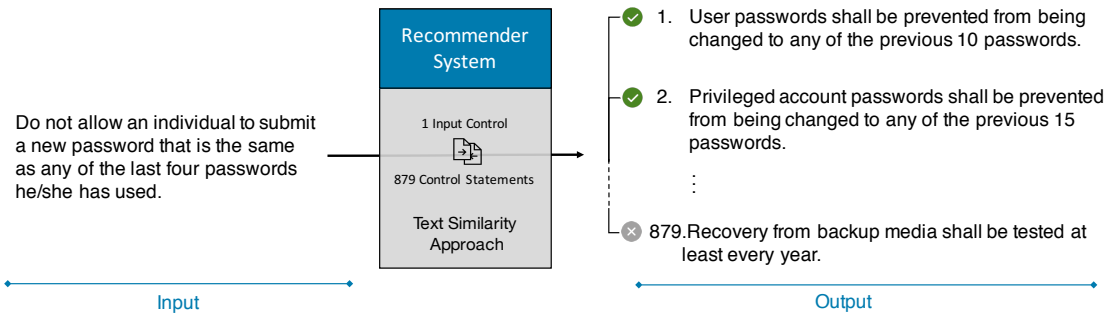


Figure 1.4.: Recommender System using a Text Similarity Approach

The recommender system accepts a control from both regulatory document or company policy. This input control is then compared with every control statement within the collection using a text similarity approach. Finally, the system sorts the collection whereby the more similar control statements appear at the top.

From an abstract technical perspective, the recommender system matches a text passage against a collection of other text passages. This view allows the system to be used in use cases of different domains and even function as a classical search algorithm.

1.2. Research Questions

During the development of the prototype of the recommender system, the thesis addresses the following six research questions:

1. To which degree does the text similarity approach solve the semantic text matching problem?

We discuss to which degree text similarity approach solve the semantic text matching problem (introduced in Chapter 2) with regard to the problem statement of *Alyne*.

2. What text similarity approach performs best - TF-IDF, Word2Vec or Doc2Vec?

We evaluate and compare the performance of three different text similarity approaches: TF-IDF, Word2Vec and Doc2Vec.

3. Which preprocessing technologies have a positive impact on the matching results?

We investigate four different preprocessing technologies including cleaning, stemming, stop-words removal and PoS tagging.

4. Does the addition of meta information to control statements improve the results?

Control statements are linked with meta information such as topic, subtopic, title and tags. We add these information to control statements and examine the impact on matching results.

5. What are good corpora to train Word2Vec and Doc2Vec?

The machine learning based text similarity approaches Word2Vec and Doc2Vec are trained on a corpus. We investigate several corpora that differ in characteristics regarding content type (off-topic/on-topic) and size (small/big) to conclude the influence on matching results.

6. Can chapter or paragraph context help to improve the results?

Chapter or paragraph context of input controls, like section titles, often contains key words that might help in the matching process. We extract context information and examine its impact.

1.3. Research Approach

This section presents our iterative research approach and introduces the ground truth as well as the quantitative criteria we use to evaluate the recommender system.

1.3.1. Iterative Approach

After literature research and the implementation of the recommender system, we perform a series of evaluations to answer the research questions (RQ) following an iterative approach.

The iteration steps are shown in Figure 1.5.

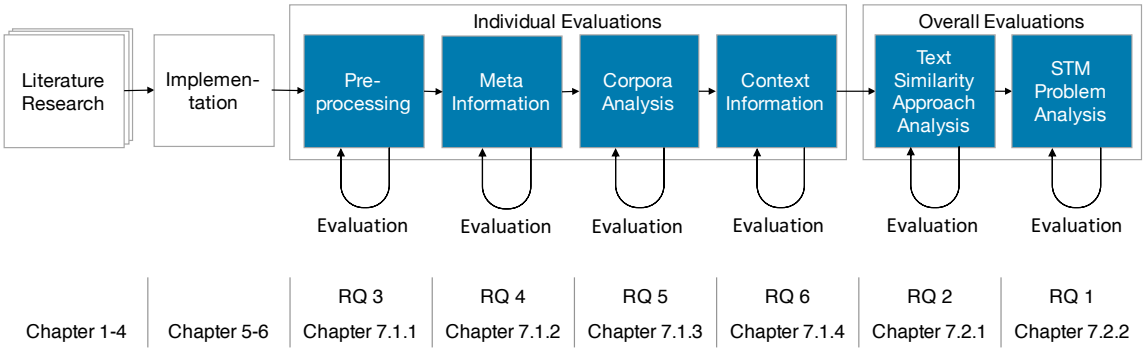


Figure 1.5.: Iterative Research Approach

First, covering research questions from 3 to 6, we perform individual evaluations on four different strategies which aim to improve matching results: preprocessing, addition of meta information to control statements, corpora analysis and addition of context information to input controls.

Second, covering research questions 1 and 2, we conduct a final overall analysis on the performance of text similarity approaches TF-IDF, Doc2Vec and Word2Vec, and assess to which degree the semantic text matching problem of regulatory documents and company policies is solved.

The iterative approach is particularly important during the individual evaluations. Due to time and hardware limitations, especially with regard to the machine learning based similarity approaches, we can only perform a subset of parameter combinations (curse of dimensionality). So, per iteration step we only change the values of one parameter, evaluate the matching results and continue with the value of the best result we identified.

We assess quantitatively the research questions from 2 to 6 with ground truth and RPS measure which are both introduced in the following two Sections. Research question 1 is evaluated qualitatively in form of a feedback sheet. The qualitative evaluation approach is described in detail in Chapter 6.2.2.

1.3.2. Ground Truth

To quantitatively evaluate the quality of the matching results, we have access to a ground truth of 1000 items from 10 different regulatory documents.

The 10 documents are listed in Table 1.2 including the number of paragraphs and number of words per paragraph.

Regulatoy Document	#Paragraphs	∅ Words/Paragraphs
BDSG[2]	24	362.88
COBIT[3] 5 2012	191	32.02
COSO[4] 2013	16	18.94
Finish Personal Data Act 523/1999	29	164.24
ISO 22301:2012	37	122.46
ISO 27001:2005	133	24.86
ISO 31000:2009	46	95.17
MAS-TRMG[5] 21	268	52.00
NIST[6] C2M2 Cyber Security Framework v1.1	95	9.02
PCI DSS[7] v3.1	161	170.16
All Ground Truths	1000	74.30
All Control Statements		21.55

Table 1.2.: Ground Truth

The ground truth, same as the control statements, are in English language.

They differ in level of detail and paragraph length. Different paragraph lengths can complicate the matching of two text passages. So, we ensure that our regulatory documents also cover extreme examples: BDSG with an average length of 363 words per paragraph is almost 17 times longer than the average of 22 for control statements. NIST with an average of 9 is more than 2 times shorter. ISO 27001 and COSO have nearly same length as the control statements.

Listing 1.1 shows an example ground truth item.

It consists of an identifier, the input control (input) and the corresponding control statements (outputs). The identifier is a unique string consisting of document title (“PCI DSS v3.1”) and paragraph (“8.2.5”) of the input control.

[2] German Federal data protection act
[3] Control Objectives for Information and Related Technologies
[4] Committee of Sponsoring Organizations of the Treadway Commission
[5] Monetary Authority of Singapore Technology - Risk Management Guidelines
[6] National Institute of Standards and Technology
[7] Payment Card Industry Data Security Standard

```
{
   "identifier": "PCI DSS v3.1 8.2.5",
   "input": "Do not allow an individual to submit a new password
      that is the same as any of the last four passwords he/she
      has used.",
   "outputs": [
      "Control_Statement_Id_00033", // "User passwords shall be
         prevented from being changed to any of the previous 10
         passwords."
      "Control_Statement_Id_00056",
      ...
   ]
}
```

Listing 1.1: Ground Truth Item

1.3.3. Evaluation Criteria

To quantify the quality of the matching results of our recommender system, we use the intuitive RP-Score (Ranking-Position-Score, RPS) [Landthaler, 2017] that averages the ranking position of the expected output contained in the ground truth.

Table 1.3 shows an examplary matching result for the ground truth item in Listing 1.1:

RP	Control Statement Id
1	**Control_Statement_Id_00033**
2	Control_Statement_Id_00544
3	**Control_Statement_Id_00056**
4	Control_Statement_Id_00612
...	(and 875 more)

Table 1.3.: Matching Result Example

The matching result is (throughout the thesis) a list of 879 control statements, while here the two correct outputs *Control_Statement_Id_00033* and *Control_Statement_Id_00056* have ranking positions 1 and 3. So, the average rank, or RP-Score, is $\frac{1+3}{2} = 2$.

The optimal RPS in this example is $(\sum_{i=1}^{\#outputs} i)/\#outputs = \frac{1+2}{2} = 1.5$.

The average optimal RPS of all ground truth items is 2.75.

1.4. Structure of the work

The remainder of this thesis is organized as follows:

- Chapter 2 gives a brief summary of the research topic of the semantic matching problem (STM) and related work.
- Chapter 3 looks at theoretical background of natural language processing, and describes preprocessing technologies and text similarity approaches that will be applied.
- Chapter 4 introduces requirements, features and implemenation details of our recommender system.
- Chapter 5 presents the platform *Citadel*, implemented in this thesis, that supports us during the evaluation of the recommender system.
- Chapter 6 describes the evaluations undertaken and presents their results, and discuss the research questions.
- Chapter 7 summarizes this thesis and discusses future work.

2. Semantic Text Matching

The matching of semantically similar text passages - throughout this thesis company policies and regulatory documents with control statements - is a common problem of different domains, in particular text-intensive areas such as compliance or law.

This chapter introduces the research topic *semantic text matching* (STM), which focuses on this problem, and looks at related research. STM is related to the area of information retrieval (IR).

2.1. Definition

According to [Landthaler, 2018], the STM problem is the detection of semantic links between text passages of one or several documents. Figure 2.1 illustrates the STM problem which is essentially an abstracted view of Figure 1.1 in Chapter 1.

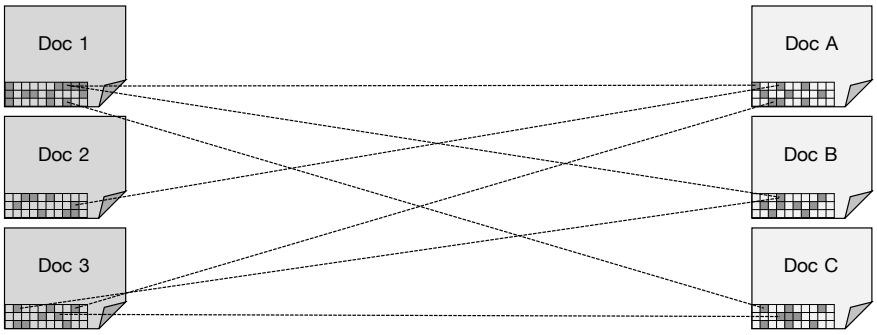


Figure 2.1.: Illustration of STM Problem, based on [Landthaler, 2018]

This paper differentiates two sub-problems of STM:

- Segmentation (not considered in this thesis): Before text passages can be matched they need to be extracted from their documents. These sources differ in structure and file formats which complicates the parsing process to retrieve relevant text passages.
- Matching (focus of this thesis): Semantically related text passages from different documents need to be matched.

2.2. Related Work

The identification of semantic links causes a lot of manual work. There have been different attempts to automate this process in several domains.

In the domain of German Tenancy Law [Landthaler, 2018] presents a vision of how to link contract clauses and legal comments using the unsupervised text similarity approach Word2Vec and TF-IDF as baseline. This paper proves the technological feasibility to some degree but also points out the challenge to distinguish very similar text passages.

In the police domain [Duan and Xu, 2016] introduce an unsupervised text similarity algorithm for 110 incidents to semantically match similar incidents. Their approach is a combination of the TF-IDF and Word Mover's Distance (WMD) [Kusner et al., 2015] which is a semantic similarity metric between text documents based on word embeddings. Results show that TF-IDF has a positive impact on the performance of WMD. TF-IDF outperforms in case types where incident descriptions are mostly the same.

[Rinott et al., 2015] propose a supervised automatic method to detect evidences from unstructured text for a given claim. [Naderi and Hirst, 2015] present a model, that combines a supervised (support-vector machine) and an unsupervised (Word2Vec) approach, to discover framing strategies for given parliamentary statements.

3. Natural Language Processing

Natural language processing (NLP) is an interdisciplinary approach in the areas of computer science, linguistics and artifical intelligence that focuses on how to make computers understand human beings by analysing and extracting the semantic meaning from natural language (English, German etc.) [Gurusamy and Kannan, 2014].

NLP provides important tools to implement our recommender system that aims to solve the STM problem. In this chapter, we introduce theoretical background regarding

- preprocessing technologies and
- the text similarity approaches TF-IDF, Word2Vec and Doc2Vec.

3.1. Preprocessing Technologies

Text preprocessing technologies aim to reduce unnecessary information and draw out important features. Preprocessing is applied to any text passage that is intended to be matched by our recommender system.

Legal documents, like regulatory documents or company policies, are generally quite clean and structured texts: They are less incomplete, noisy and inconsistent than other real world instances such as Twitter tweets that contain colloquial language and spelling mistakes. This fact already simplifies but does not replace preprocessing, for example, we cannot assume all sources to be clean, some controls are short bullet points and others are long sentences, and special characters are hard to interpret.

Figure 3.1 shows the pipeline of common preprocessing techniques which we implement (Chapter 4) and investigate (Chapter 6) for our recommender system. These include tokenization & cleaning, stopwords removal, stemming and part-of-speech (PoS) tagging.

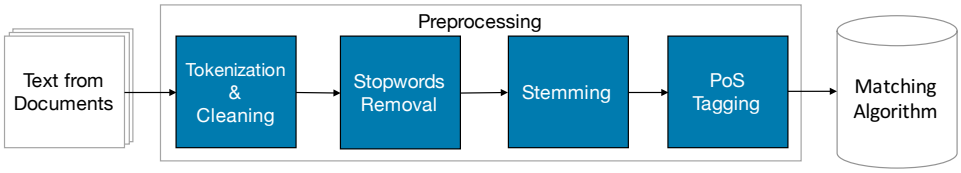


Figure 3.1.: Preprocessing Pipeline, based on [Vijayarani and Ilamathi, 2015]

We read about state-of-the-art preprocessing technologies in [Gurusamy and Kannan, 2014] and [Vijayarani and Ilamathi, 2015]:

Tokenization & Cleaning: Tokenization is the process that splits a text into its individual components (tokens) such as words, symbols or punctuations. We then can apply different techniques in order to clean these tokens. Commonly, digits, punctuation and symbols (hash tags, slashes etc.) are removed and words converted to lowercase.

Stopwords Removal: Stopwords removal is the process that removes filler words that appear frequently but give little or no meaning to the text like, for example, articles or prepositions. These words are typically defined in a pre-compiled list and can be either general or domain-specific. This process removes noise and saves memory as well as processing time by shrinking the vocabulary space. It is noted that incorrectly identified stopwords can remove relevant information.

Stemming: Stemming is the process that transforms a word to its stem by removing its suffix. For example, prevented, preventing, prevention result all in prevent. It aims to create accurate matching pairs and aggregate key words.

PoS Tagging: Part-of-speech (PoS) tagging marks up each word of a sentence with its word class by labelling nouns, verbs, adjectives etc. We use this technique to identify and then remove word classes that contains more noise than information - it is known that nouns usually hold most of the semantic information of a sentence. The Penn Treebank lists available part-of-speech tags[1].

Table 3.1 shows an exemplary application of the introduced preprocessing technologies to the control statement from Listing 1.1.

Preprocessing	Result
-	“User passwords shall be prevented from being changed to any of the previous 10 passwords.”
Tokenization & Cleaning	[“user”, “passwords”, “shall”, “be”, “prevented”, “from”, “being”, “changed”, “to”, “any”, “of”, “the”, “previous”, “passwords”]
Stopwords Removal	[“user”, “passwords”, “prevented”, “changed”, “previous”, “password”]
Stemming	[“user”, “password”, “prevent”, “chang”, “previous”, “password”]
PoS Tagging (Nouns/Verbs)	[“user”, “password”, “prevent”, “chang”, “password”]
PoS Tagging (Nouns)	[“user”, “password”, “password”]

Table 3.1.: Exemplary Application of Preprocessing Technologies

[1] `https://www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html`, last accessed February 2018

3.2. Text Similarity Approaches

A text similarity approach creates a vector representation of a text passage and maps the vector into a vector space model. When, then, two text passages are compared, their semantic similarity is determined by calculating the distance between their vectors using text similarity measures like, throughout the thesis, the cosine distance. [Salton et al., 1975] A summary of similarity measures can be found in [Gomaa and Fahmy, 2013]. The similarity (or distance) is reflected by a number between -1 and 1 where 1 indicates high similarity.

We introduce the three approaches which the recommender system implements:

- TF-IDF as a very simple but efficient algorithm and
- two machine learning based approaches,
  - Word2Vec (word embeddings) and
  - Doc2Vec (document embeddings).

3.2.1. Term Frequency - Inverse Document Frequency

Term frequency - inverse document frequency, abbreviated TF-IDF, is a term-weighting system [Salton and Buckley, 1988] that expresses the relevance of a term t for a document D (text passage) in a collection. It is the product of term frequency (tf) and inverse document frequency (idf).

Concept

The term frequency specifies how often a term occurs in a document in relation to the document size. The algorithm assumes that frequently occurring terms are key words and they thus have a higher weight than others. In our example from Listing 1.1, the terms *passwords* and *be* occur 2 times, and the document size is 15 words. The tf value of both terms is

$tf(t, D) = tf(passwords, D) = tf(be, D) = \frac{2}{15} = 0.13.$

The inverse document frequency represents in how many documents a term appears. When a term occurs in several documents, like stopwords do, it might contain less information and thus has a lower weight. In our example, the term *passwords* occur 25 and *be* 100 times with a collection size of 879 control statements. The idf values are

$idf(passwords) = \log \frac{879}{25} = 1.55$ and $idf(be) = \log \frac{879}{100} = 0.94.$

The final TF-IDF weights are $tfidf(passwords, D) = 0.13 * 1.55 = 0.20$ and $tfidf(be, D) = 0.13 * 0.94 = 0.12$. So, the key word *passwords* has a higher weight than word *be*.

From Text Passage to Vector

It is now necessary to clarify, based on the concept of TF-IDF, how to transform a text passage to a vector in order to finally calculate similarities.

The transformation is described in pseudo-code in Algorithm 1.

```
   input  : Text text, TFIDFVocabCache cache
   output: Vector[] vector
1  words ← Tokenize(text)
2  vector ← InitialiseVector(Count(cache))
3  for word in words do
4  |   get cache index for word, calculate TF-IDF value and put value to vector
5  |   index ← IndexOf(word,cache)
6  |   vector[index] ← CalculateTFIDF(word)
7  end
```

Algorithm 1: TF-IDF - from Text to Vector

The algorithm accepts two parameters (input), a text and a vocabulary cache which contains all terms that appeared within the collection. It returns the final vector representation of the text passage (output).

The text is first tokenized to single words and the final vector is initialized with a length which is equal to the (distinct) number of words of the cache. Second, the words are iterated. For each word, the algorithm reads the cache index and calculates the TF-IDF value. This value is then put into the vector on the position of the cache index. Finally, the vector is returned.

Because the algorithm can only calculate TF-IDF values for words appearing in the collection, other unknown words are ignored by the algorithm.

3.2.2. Word2Vec (Word Embeddings)

Word embeddings are words represented as vectors of real numbers while the distance of two word vectors indicates semantic similarity.

[Mikolov et al., 2013a, Mikolov et al., 2013b] proposes an unsupervised machine learning algorithm, called *Word2Vec*, that uses a neural network to efficiently train billions of word representations in a vector space. These papers show that the algorithm can capture (to some degree) syntactic analogies (like “secure”, “securely” or “developer”, “development”) as well as semantic analogies, like “the country to capital city relationship”. It is also noted that simple algebraic operations can result in semantically meaningful results: a very famous example is $King - Man + Woman = Queen$.

Concept

The basic assumption of Word2Vec is that words in a corpus are considered as related when they appear in the same context, which means their surrounding words are similar.

Figure 3.2 shows two different architectures of Word2Vec to create word embeddings: the continuous bag-of-words model (CBOW) predicts a word based on its context and the continuous skip-gram model predicts the context for a given word.

(a) CBOW (b) Skip-gram

Figure 3.2.: Word2Vec models, based on [Mikolov et al., 2013a, Le and Mikolov, 2014]

Our recommender system implements the Skip-gram model because it generally performs better than CBOW, according to [Mikolov et al., 2013a].

Word2Vec provides several parameters to optimise its vectors. We only introduce two that appear in the further context:

- Vector size: Length of word vector, a bigger size means higher memory usage but also more space to numerically describe a word
- Epochs: Number of iterations over whole training corpus during training

From Text Passage to Vector

The concept of *Word2Vec*, as the name indicates, is based on word level so that we first need to transfer this approach to text level. Therefore, we simply average word vectors.

The transformation from text passage to vector is shown in pseudo-code in Algorithm 2.

The algorithm accepts two parameters (input), a text and a vocabulary cache which stores all words appeared within the corpus and their word vectors. It returns the final vector representation of the text passage (output).

```
   input : Text text, WordEmbeddingsVocabCache cache
   output: Vector[] vector
 1 words ← Tokenize(text)
 2 matrix ← InitialiseMatrix(Count(words), GetWordVectorSize (cache))
 3 i ← 0
 4 for word in words do
 5 |   get word vector from cache and put to matrix
 6 |   matrix[i] ← GetWordVector(word, cache)
 7 |   i ++
 8 end
 9 average rows of matrix
10 vector ← Average(matrix)
```

Algorithm 2: Word Embeddings - from Text to Vector

The text is first tokenized to single words and we initialise a matrix with the word vector size and the number of words in the text passage. Second, we iterate the words and add the corresponding vectors to the matrix. Finally, the matrix is averaged and the resulting vector is returned.

3.2.3. Doc2Vec (Document Embeddings)

Document embeddings are documents (text passages) represented as vectors of real numbers while the distance of two document vectors indicates semantic similarity.

[Le and Mikolov, 2014] propose the framework *Paragraph Vectors* (PV) that can create vector representations from text of any length, such as sentences or documents. The unsupervised algorithm, called *Doc2Vec*, is an extension of Word2Vec and algebraic operations can here also result in semantically meaningful results [Dai et al., 2015].

Figure 3.3 shows two different models of Doc2Vec, the distributed memory model (PV-DM) and the distributed bag of words (DBOW).

PV-DM is similar to CBOW with the difference that an additional paragraph token (P-ID) is added to the context. This token acts as the memory or topic of the current context. Together they predict the next word. It is noted that this model considers the word order.

DBOW is similar to Skip-gram with the difference that a paragraph token instead of a word predicts its context. The word order is hereby not considered.

(a) PV-DM (b) PV-DBOW

Figure 3.3.: Doc2Vec models, based on [Le and Mikolov, 2014]

Although [Le and Mikolov, 2014] state a better performance for PV-DM, our recommender system implements PV-DBOW because our initial evaluations, and also [Lau and Baldwin, 2016], show contrary results.

4. Implementation Recommender System

In this section we present implementation details of the recommender system that uses text similarity approaches to semantically match a text passage against a collection of other text passages or, transferred to our business problem, an input control against control statements.

We first summarize functional and non-functional requirements which the recommender system addresses and implements (4.1). Second, the conceptual system structure is shown consisting of matching algorithm, its configuration and evaluation (4.2).

The matching algorithm is referred to hereinafter as *Matcher*.

4.1. Requirements

At the beginning of the thesis, functional and non-functional requirements for the recommender system are defined.

The following functional requirements are addressed:

- The system shall provide an interface to configure all relevant parameters of a Matcher, including parameters of the underlying text similarity approaches (4.2.1)
- The configuration shall be done in writing a configuration file or programmatically (4.2.1)
- The Matcher shall calculate the similarity between two text passages (4.2.2)
- The Matcher shall recommend text passages from a defined collection that are semantically related to a given search query (4.2.2)
- The system shall save a Matcher instance into a file and load it from a file (4.2.2)
- The system shall evaluate a Matcher instance by using ground truths (4.2.3)

And the following non-functional requirements are addressed:

- The system shall be implemented in the programming language Java: In the future it is intended to be integrated into the already existing JAVA web application *Lexia* [Waltl, 2015], a data science environment for legal texts.
- The system shall be implemented using the framework DeepLearning4J (DL4J): It is an "open-source, distributed, deep learning library for the JVM"[1]. The framework includes natural language processing tools as well as implementations of the text similarity approaches TF-IDF, Word2Vec and Doc2Vec.

4.2. System Structure

The recommender system consits of three components: Configuration, Matcher and Data Sets. We can see the conceptual system structure in Figure 4.1.

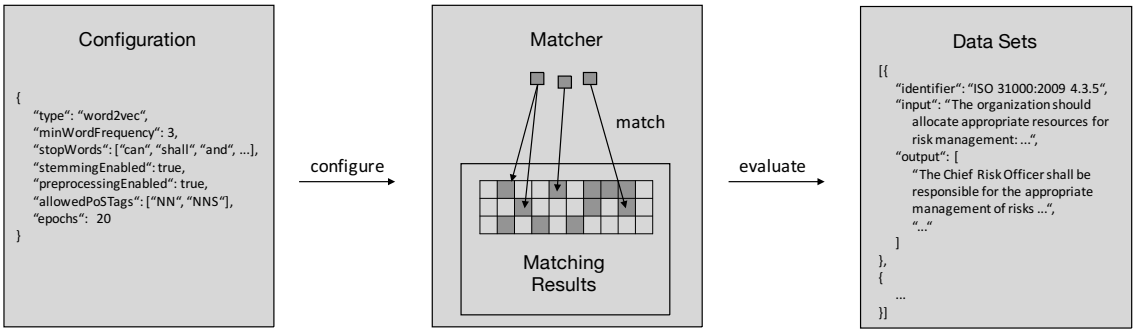


Figure 4.1.: System Components

We first specify the text similarity approach and define its set of configuration paramters. Second, a Matcher is instantiated accordingly and then ready to match related text passages. Finally, the performance of the Matcher can be evaluated by data sets that contains ground truths.

4.2.1. Configuration

The component *Configuration* encapsulates all functionality needed to configure relevant parameters of a Matcher instance including parameters of the underlying text similarity approach.

[1] `https://deeplearning4j.org/index.html`, last accessed February 2018

The conceptual structure of *Configuration* is shown in the class diagram in Figure 4.2.

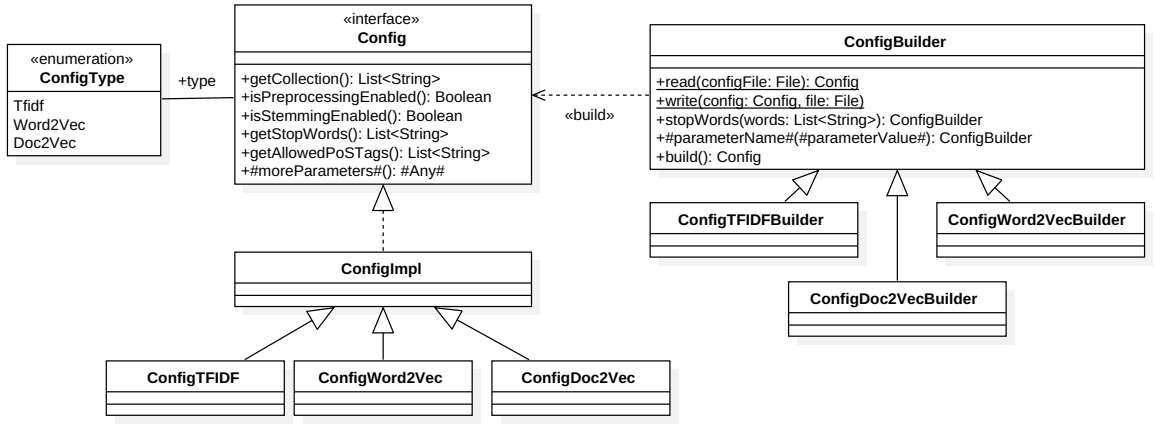


Figure 4.2.: Configuration Class Diagram

The interface *Config* and the class *ConfigBuilder* as well as its subclasses are public.

The *Config* interface provides getter methods to access the type of text similarity approach (*ConfigType*), preprocessing settings and approach-specific parameters as well as the collection of text passages that are aimed to be matched. *ConfigImpl* implements the interface and ensures - together with its subclasses *ConfigTFIDF*, *ConfigWord2Vec* and *ConfigDoc2Vec* - that only approach-related values are stored. An instance of *Config* can exclusively be built by a *ConfigBuilder*.

The *ConfigBuilder* is an abstract class that allows to create a *Config* instance programmatically or by loading a configuration from a file. The concrete subclasses *TFIDFConfigBuilder*, *Word2VecConfigBuilder* and *Doc2VecConfigBuilder* provide setter methods for all approach-specific parameters. Method *build* finalizes the building process and returns a *Config* instance.

The static methods *read* and *write* enables to load and save a configuration file in JSON format, as shown in Figure 4.3.

```
{
  "type": "tfidf",
  "stemmingEnabled": true,
  "allowedPoSTags": [ "NN", "NNS" ],
  ...
}
```

Figure 4.3.: Example Configuration File

4.2.2. Matcher

The component *Matcher* is responsible to match a text passage against a collection of other text passages - the core function of the recommender system.

The conceptual structure of *Matcher* is shown in the class diagram in Figure 4.4.

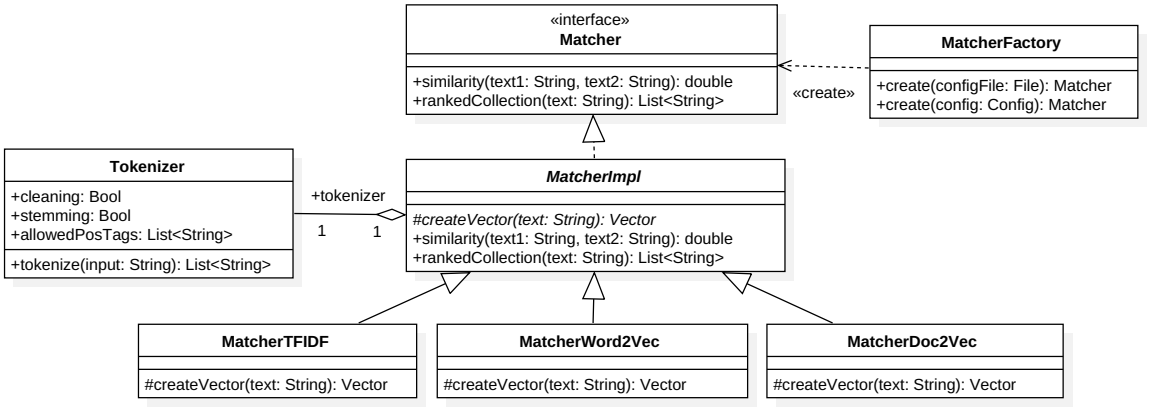


Figure 4.4.: Matcher Class Diagram

The interface *Matcher* and the class *MatcherFactory* are public.

The *MatcherFactory* allows to create a *MatcherImpl* object by specifying the configuration file or directly by an instance of *Config*. Depending on the passed *ConfigType*, either *MatcherTFIDF*, *MatcherWord2Vec* or *MatcherDoc2Vec* is instantiated and finally configured.

The *MatcherImpl* implements the interface *Matcher* which provides the functionality of the matching algorithm: The method *rankedCollection* returns the collection sorted by the similarities with the input text passage - related passages at the top.

The similarity value for each collection item is calculated by the method *similarity* which takes two strings (item and input text passage), transforms both into a vector representation and applies the Cosine distance. The transformation from string to vector is realised by method *createVector* that is basically the only piece of code in which the approach-specific subclasses *MatcherTFIDF*, *MatcherWord2Vec* and *MatcherDoc2Vec* differ. The theoretical background about string-to-vector transformation is explained in Section 3.2.

Before a text passage is transformed to a vector, however, the class *Tokenizer* splits the string into words. Depending on the configuration, preprocessing technologies are then applied, in particular cleaning, stemming and PoS tagging.

4.2.3. Data Sets

The component Data Sets is responsible for a quantitative assessment of a Matcher instance and provides the final evaluation results presented in chapter 6.

The conceptual structure of *Data Sets* is shown in the class diagram in Figure 4.5.

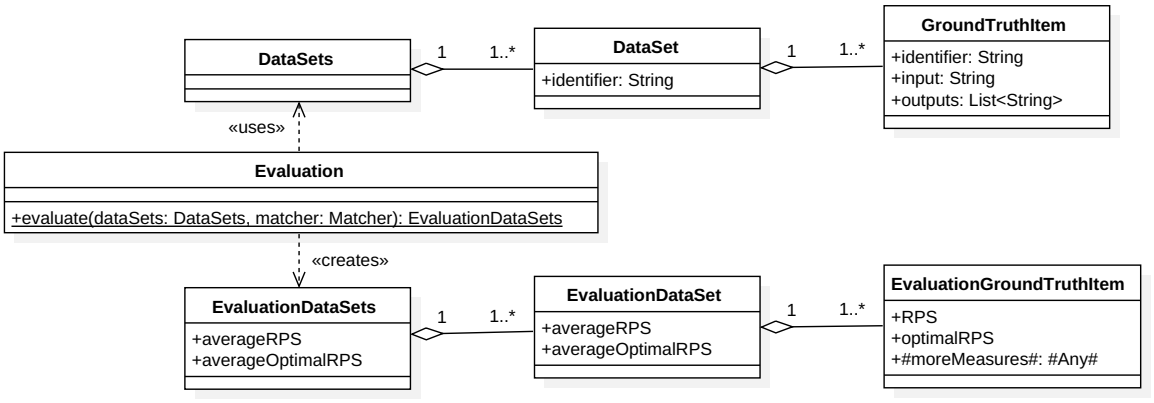


Figure 4.5.: Data Sets Class Diagram

All classes are publicly visible.

The classes *DataSets*, *DataSet* and *GroundTruthItem* store ground truths information. *DataSets* groups a set of *DataSet* objects and these group a set of *GroundTruthItem* objects. For example, “BSDG”, “COBIT 5” etc. from Table 1.2 correspond to *DataSet* instances, Listing 1.1 to an instance of *GroundTruthItem*.

The class *Evaluation* provides the core method *evaluate*. It accepts two arguments, a *DataSets* and a Matcher instance, runs the matching process and finally assesses the results by calculating relevant numbers, such as RPS or optimal RPS (introduced in Section 1.3.3), based on the passed ground truths. As a result, an instance of *EvaluationDataSets* is returned.

The classes *EvaluationDataSets*, *EvaluationDataSet* and *EvaluationGroundTruthItem* contains evaluation data related with *DataSets*, *DataSet* and *GroundTruthItem*.

5. Implementation GUI Citadel

Citadel is a platform that enables to easily manage and evaluate Matcher instances that are created by the recommender system.

The aim is to provide a graphical user interface that supports us throughout the entire life cycle of a Matcher instance, and in particular to assist regulatory experts from *Alyne* while analyzing matching results.

The assistance of regulatory experts is the main motivation to build such platform. The experts are potential business users of the recommender system functionality who strongly supported us during the work on this thesis: They understand their domain, know vocabulary and wording of relevant documents, so that they evaluate matching results from a different point of view and thus give valuable feedback. Because the experts are usually non-IT professionals who might struggle with console output, we provide a more advanced interface.

This chapter presents the system setup (5.1) of *Citadel* and introduces its core features (5.2).

5.1. Setup

The platform is designed to run on a remote server that operates day and night. This setup decision is mainly driven by two requirements: We need, on the one hand, constant availability to enable experts permanent access and, on the other hand, powerful hardware to create Matchers which are based on CPU and memory intensive machine learning algorithm. Both facts exclude the use of a local personal computer.

Therefore, *Citadel* is deployed on an Amazon EC2 Linux Instance with 32 GB of memory which is accessible from the outside.

The setup follows a 3-tier-architecture consisting of backend *CitadelServer*, frontend *CitadelClient* and database *CitadelDB*. The system structure of these three components is shown in the deployment diagram in Figure 5.1.

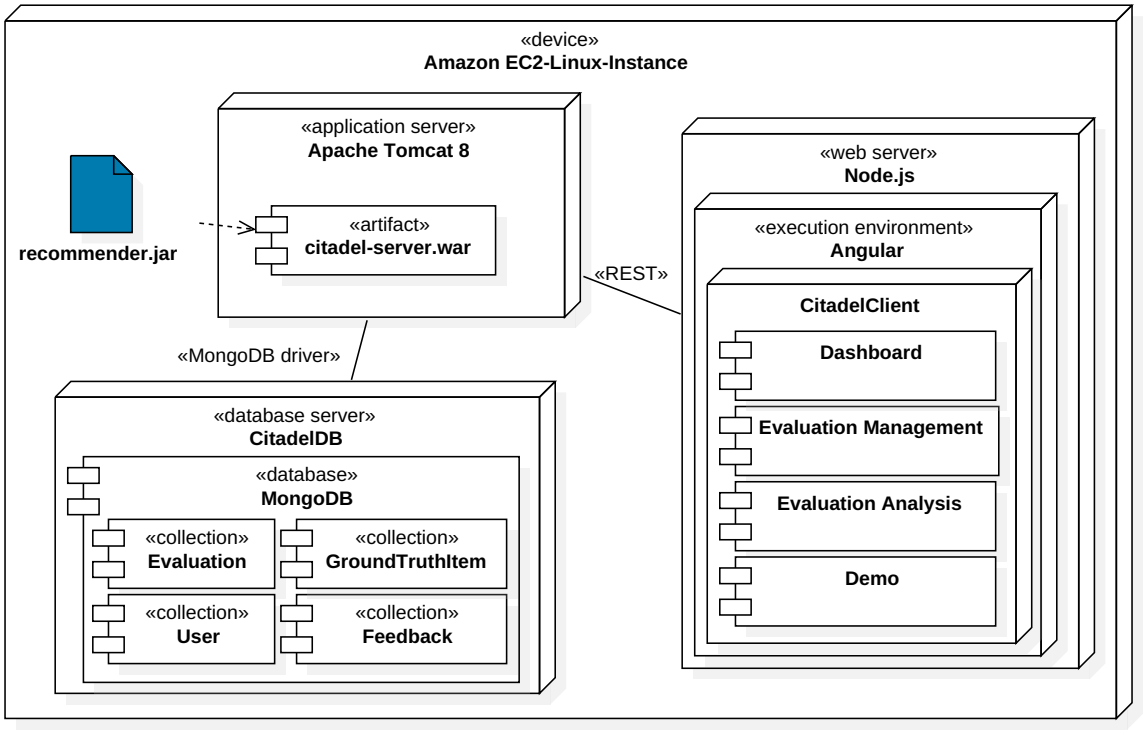


Figure 5.1.: Deployment Diagram of Citadel

The backend *CitadelServer* is a Java EE application[1] which runs on an Apache Tomcat 8[2]. The application implements the business logic of the platform which is accessible via a REST API: It consumes the functionality of the integrated recommender system (*recommender.jar*), introduced in the previous chapter, and provides, among others, an interface to start evaluations and request their matching results as well as statistics. Moreover, because the server is online, user authentication is implemented.

The database *CitadelDB* is a document-oriented MongoDB[3] and is connected to the backend via MongoDB Java Driver[4]. It stores evaluation data including all ground truth items as well as feedback information provided by users, in particular regulatory experts.

The frontend *CitadelClient* is an Angular 5 application[5] and is connected to the backend via a REST-API. The core features include *Dashboard*, *Evaluation Management*, *Evaluation Analysis* and *Demo* which are explained in detail in the following section.

[1] `https://docs.oracle.com/javaee/6/`, last accessed February 2018
[2] `https://tomcat.apache.org/download-80.cgi`, last accessed February 2018
[3] `https://www.mongodb.com`, last accessed February 2018
[4] `https://mongodb.github.io/mongo-java-driver/`, last accessed February 2018
[5] `https://angular.io`, last accessed February 2018

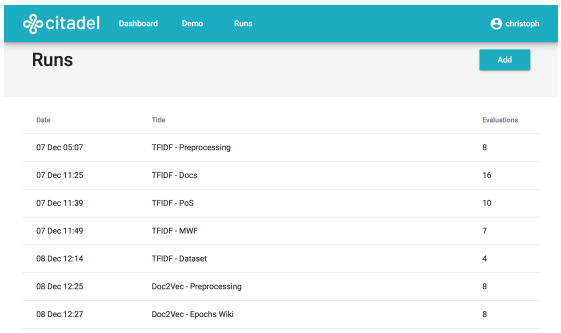
5.2. Features

The core features of *Citadel* include the management and the analysis of Matcher evaluations.

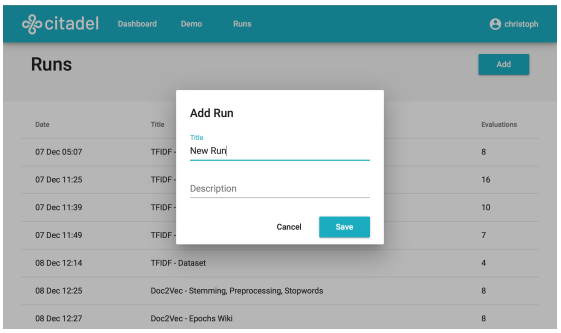
5.2.1. Evaluation Management

The evaluation management provides a convenient interface to create, start and monitor the evaluation process as well as finally view the evaluation results.

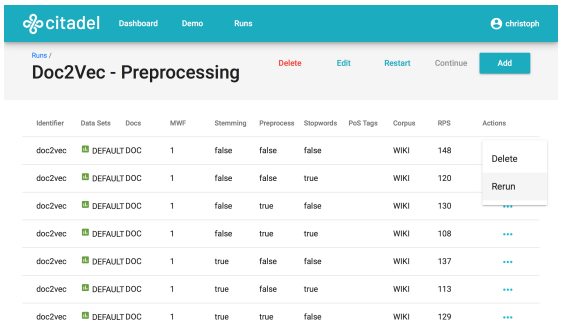
Figure 5.2 shows four relevant screenshots of evaluation management.



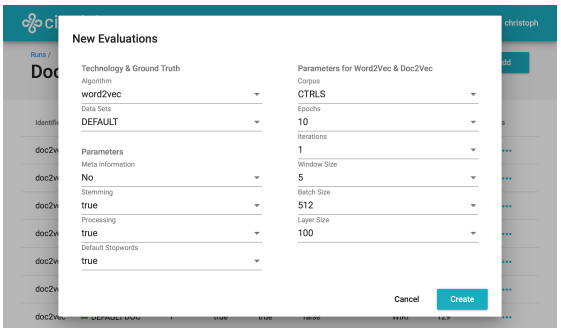
(a) Runs



(b) Run Creation



(c) Run and its Evaluations



(d) Evaluation Creation

Figure 5.2.: Run and Evaluation Management

Due to a high number of performed evaluations during this thesis, we group related evaluations to so-called runs to ensure a good overview, as shown in Figure 5.2(a). For example, run “Doc2Vec - Preprocessing” contains 8 evaluations that combines different preprocessing techniques applied to text similarity approach Doc2Vec. A run can be created by providing a title and an optional description, as shown in Figure 5.2(b).

The detail page of a run lists already existing Matcher evaluations, as shown in Figure 5.2(c). It also provides several controls to edit the title of the run, to delete the entire run and to restart one or every Matcher (necessary when Matcher implementation has changed).

And last but not least, there is a dialog that enables the creation of evaluations, as shown in Figure 5.2(d). It provides drop-down lists to specify the text similarity approach, preprocessing technologies and other approach-specific parameters, whereby multi selection is allowed. For example, when you select the approaches “Doc2Vec”, “Word2Vec” and “TF-IDF”, and stemming “Yes” and “No”, the system starts 6 evaluations.

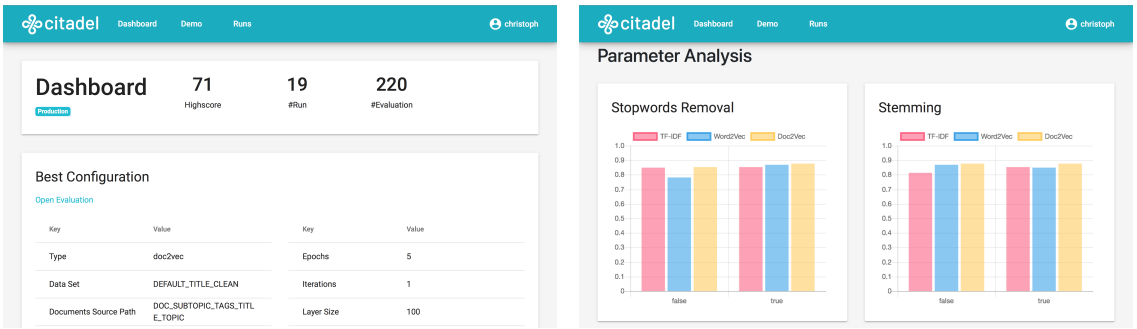
5.2.2. Evaluation Analysis

Besides management, *Citadel* also provides three screens that aims to view and analyze Matcher evaluations: dashboard, evaluation detail screen and demo.

Dashboard

The dashboard screen summarizes and aggregates results from all Matcher evaluations that are performed during this thesis.

Figure 5.3 shows two screenshots of the dashboard.



(a) Summary and Best Configuration

(b) Parameter Analysis

Figure 5.3.: Dashboard

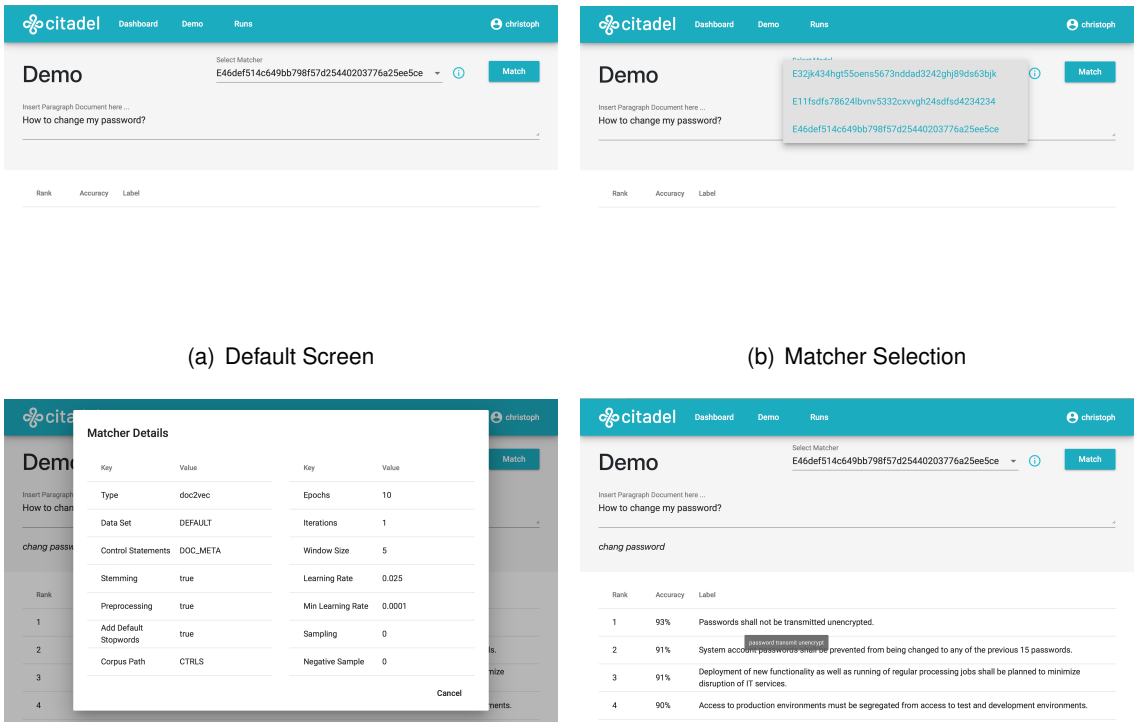
At the top of the dashboard screen, aggregated key facts like total number of evaluations or the currently highest RPS value of the best performing Matcher including its configuration parameters are listed, as shown in Figure 5.3(a).

Another dashboard section is the parameter analysis. It provides charts for relevant configuration parameters and visualizes their impact on RPS distinguished by text similarity approaches. For example, the chart *Stopwords Removal* in Figure 5.3(b) shows that the elimination of stopwords has a positive effect in particular on approach Word2Vec.

Demo

The demo screen provides an interface that allows to test dynamically and interactively the matching behavior of an already created Matcher instance.

Figure 5.4 shows four screenshots related with the demo.



(a) Default Screen

(b) Matcher Selection

(c) Matcher Details

(d) Matching Results

Figure 5.4.: Demo

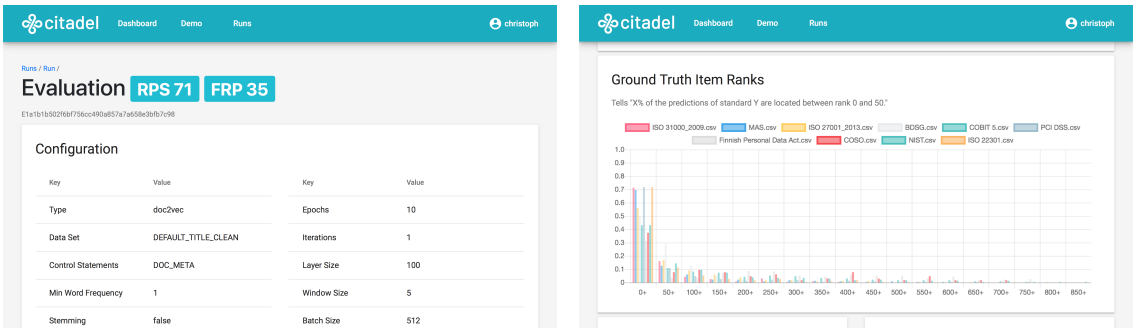
Figure 5.4(a) shows the demo screen before any matchings have happened. This screen provides a text field to enter a control, or any other strings like “How to change my password?”, which shall be matched. You can specify the Matcher by selecting the corresponding instance in the drop-down list shown in Figure 5.4(b). By tapping the info button you can view a dialog containing the configuration parameters of the selected Matcher, as shown in Figure 5.4(c). And the button “Match” triggers the matching process.

Figure 5.4(d) shows the demo screen after a matching has performed. A list view presents the matching result which has already been sorted by their similarity with the input. You can see the preprocessed input below the text field and the preprocessed matching items by hovering over the list, those simplified texts that were actually matched.

Evaluation Details and Feedback Loop

The evaluation detail screen provides detail information on individual Matcher evaluations and enables a feedback loop between experts.

Figure 5.5 shows two screenshots related with evaluation details.

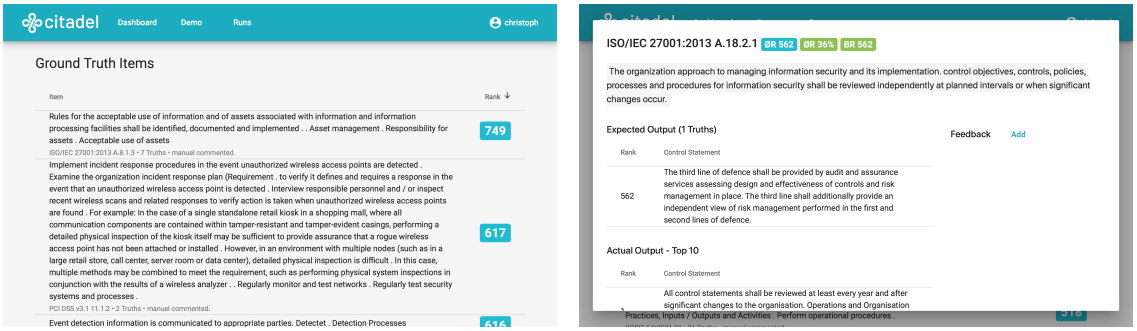


(a) RPS and Configuration Parameters (b) Various Charts

Figure 5.5.: Evaluation Details

At the top of the detail screen, you read the RPS value and configuration parameters of an evaluation, as shown in Figure 5.5(a). Moreover, aggregated data is visualized in form of plots, like the chart titled “Ground Truth Item Ranks” from Figure 5.5(b): it illustrates the distribution of the ground truth items, and we can see that most items are ranked better than a RPS of 50.

Figure 5.6 shows two screenshots related with the analysis of ground truth items.



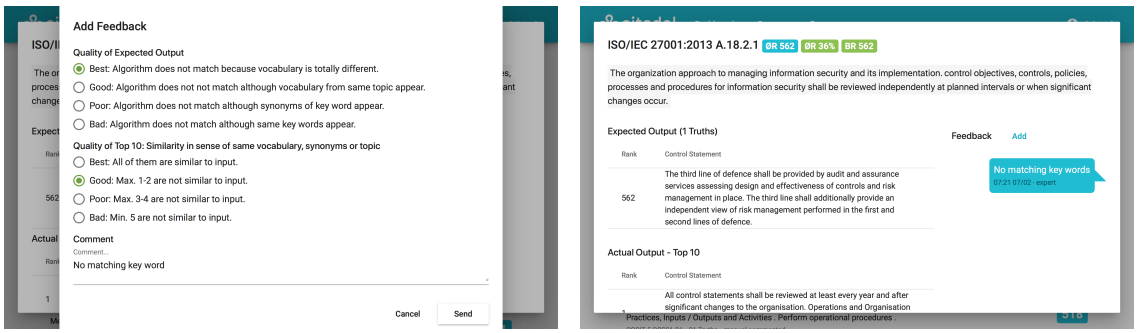
(a) Ranks of Ground Truth Items (b) Expected and Actual Outputs of Ground Truth Item

Figure 5.6.: Details of Ground Truth Item

For a thorough analysis, experts get an insight into concrete matching results by examining the list view at the end of the screen in Figure 5.6(a). A list item contains the input and the reached RPS of a ground truth item. To quickly find best and worst results the list provides a sorting function.

Tapping a list item opens a more detailed view of the selected ground truth item which is shown in Figure 5.6(b): In addition to input and RPS mentioned above, this view also presents the ranking of expected outputs and the actual top 10 matching results. This figure demonstrates a negative example in which the expected control statements are ranked by positions 179 and 277 instead of the ideal ranking of 1 and 2. By tapping the “Add” button, next to the label “Feedback”, you can access the feedback sheet.

Figure 5.6 shows two screenshots related with the feedback loop.



(a) Feedback Sheet (b) Submitted Feedback

Figure 5.7.: Feedback Loop

The feedback sheet is shown in Figure 5.7(a). This sheet is particularly intended to be used by experts to submit structured and documented feedback. It contains two multiple choice questions regarding relatedness between input and expected outputs as well as input and top 10 list. The text field allows to provide free comments.

The submitted feedback including its creation date and the username of the author is stored in the database and accessible at any time, as in Figure 5.7(b).

6. Evaluation

This chapter evaluates our recommender system and investigates how far text similarity approaches - TF-IDF, Word2Vec and Doc2Vec - solve the STM problem of company policies and regulatory documents. Here we answer the research questions of this thesis.

First, we iteratively perform four individual evaluations that test strategies which aim to support the Matcher (6.1). Second, an overall evaluation is conducted that examines the final matching results with regard to the text similarity approaches and *Alyne*'s business problem (6.2).

During the evaluation, our platform *Citadel* supports us to manage Matcher instances and to get insights into results. We create and test a total of 278 different Matchers of which 75 instances and their results are directly included in this work.

Initially, the performance of a Matcher that has not been preprocessed or optimized is tested. We start to train word and document embeddings on an on-topic corpus with a size of 200 MB (see WIKI_TOPIC in Section 6.1.3), 5 epochs, vector size of 100 and other standard parameters set by DL4J.

The results of the initial evaluation are shown in Figure 6.1.

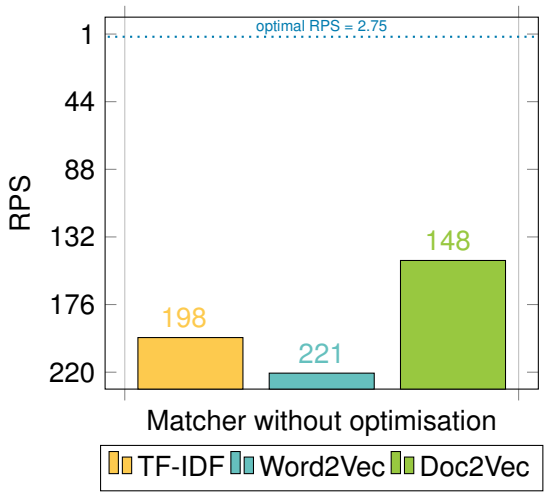


Figure 6.1.: Initial Evaluation

The plot shows the reached RPS for each approach - TF-IDF, Word2Vec and Doc2Vec. Although the score could theoretically be any value in between 2.75 (= optimal RPS, see Section 1.3.3) and 879 (= number of control statements), practically, however, the score varies in

the upper quarter. So, to make differences in rankings more clear, we start the visualization from position 220.

The RPS for TF-IDF is initially 198, for Word2Vec 221 and for Doc2Vec 148. Due to the fact that we have not performed any optimization at this point, we only note that the approaches differ considerably and the optimal RPS is far away.

6.1. Individual Evaluation

Starting from the results of the initial evaluation, we now iteratively apply four strategies that aim to improve the matching results. These include

- application of preprocessing techniques (research question 3),
- addition of meta information to control statements (research question 4),
- corpora analysis (research question 5) and
- addition of context information to input controls (research question 6).

6.1.1. Preprocessing Analysis

This first iteration applies the four preprocessing techniques introduced in Section 3.1 that includes cleaning, stopwords removal, stemming and PoS tagging, and evaluates their impact on the matching results.

Preparation

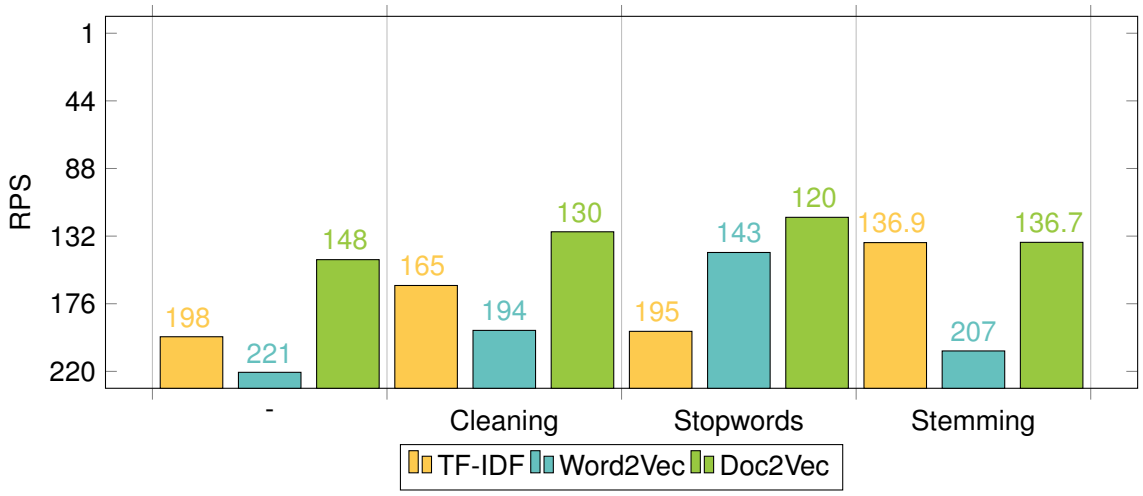
We create 12 Matchers per text similarity approach: 8 Matchers test every combination of cleaning, stopwords removal and stemming, and another 4 evaluate different PoS tags combinations consisting of nouns (N), nouns and verbs (NV), nouns and adjectives/adverbs (NA), and nouns, verbs and adjectives/adverbs (NVA).

All texts are cleaned by lowercasing and removing line breaks as well as non-alpha characters. We eliminate general stopwords from a pre-compiled list provided by DL4J. The Penn treebank tags used for nouns are NN, NNS, NNP, NNPS, for verbs VB, VBD, VBG, VBN, VBP, VBZ, and for adjectives/adverbs JJ, JJR, JJS, RB, RBR, RBS.

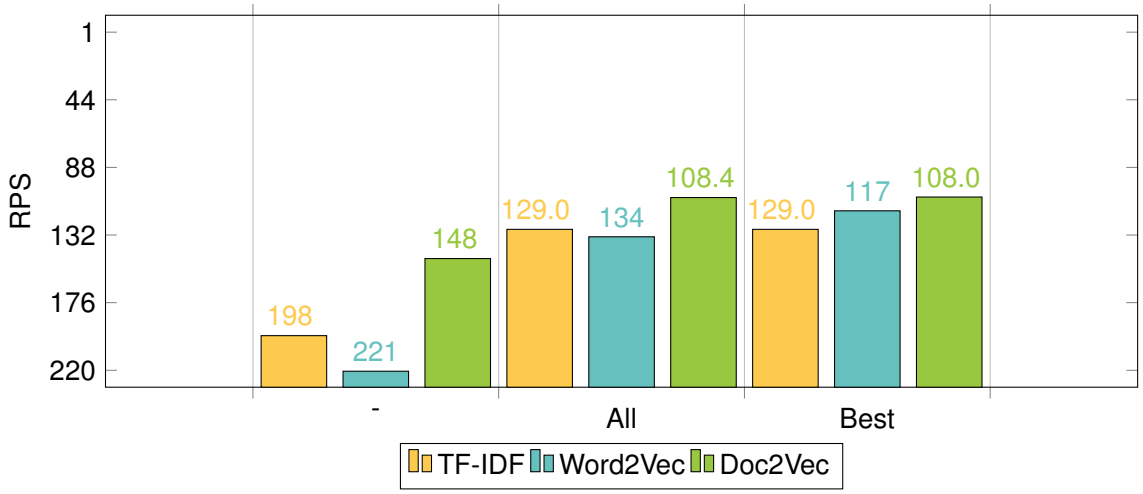
Results

The results show that all of these four preprocessing techniques support the Matcher and together significantly improve the matching results. The RPS of TF-IDF is increased from 198 to 129 (+69 positions), Word2Vec from 221 to 114 (+107) and Doc2Vec from 148 to 107 (+41).

The impact of cleaning, stemming and stopwords removal is illustrated in Figure 6.2 which is subdivided in a separated application of these techniques in Figure 6.2(a) and a combined application in Figure 6.2(b).



(a) Separate Application



(b) Combined Application

Figure 6.2.: Cleaning, Stopword Removal and Stemming

Cleaning and stemming have a positive effect particularly for TF-IDF (+33 and +60, Word2Vec +27/+18 and Doc2Vec +14/+9). Both techniques create more accurately matching terms which is a fundamental requirement that TF-IDF functions properly. Word2Vec and Doc2Vec, on

the other hand, might treat unclean or not stemmed terms like synonyms which allow indirect matching.

Stopwords removal has a very positive impact on Word2Vec and Doc2Vec (+78/+28), greater than on TFIDF (+3). While Word2Vec creates a text vector by just summing up word vectors, the IDF of TFIDF gives lower weights to frequently occuring terms - such as stopwords - which leads to a reduced impact on the final text vector and thus to better results.

A combined application of all preprocessing techniques - cleaning, stopwords and stemming - leads to clear improvements compared to a separated application. Applying all techniques creates the best matching results for TFIDF (+69). For Word2Vec and Doc2Vec, however, the Matcher that uses cleaning, stopwords removal and *not* stemming gets the best results (+91/+40).

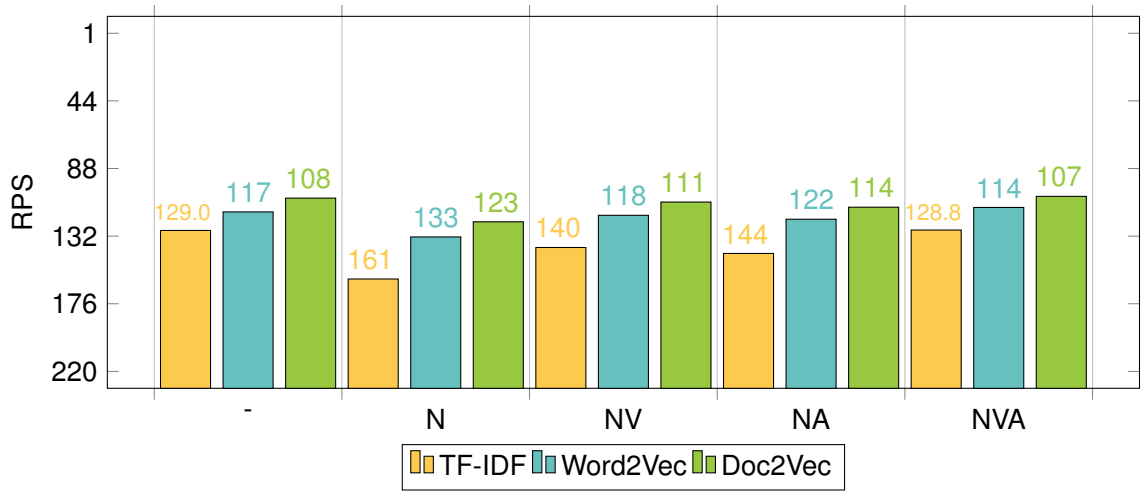


Figure 6.3.: PoS Tagging

The impact of PoS tagging is positive but low for all three approaches, as shown in figure 6.3. Tag NVA gives the best and the only positive results for TF-IDF (+0), Word2Vec (+3) and Doc2Vec (+1). Tags N, NA and NV cause negative effects. The PoS tagging iteration shows that further text cleaning can improve the matching process, but the risk exists that valuable information are considered as noise.

In the next iterations, the TF-IDF approach applies cleaning, stopwords removal and stemming. Word2Vec and Doc2Vec approaches continue with cleaning and stopwords removal but do not use stemming. PoS Tagging is not applied due to slight improvements and increased computing times.

6.1.2. Meta Information

The second iteration extends control statements by key words and matches then the extended statements against input controls. Adding meta information can create new matching pairs which boost up related controls statements and thus automatically boost down unrelated ones.

Preparation

The *Alyne* database provides meta information for each control statement. Listing 6.1 presents an example control statement with additional fields including topic, subtopic, title and tags.

```
{
   "id":        "Control_Statement_Id_00033",
   "topic":     "Password Management",
   "subTopic":  "Password History",
   "title":     "User Password History Length",
   "statement": "User passwords shall be prevented from being
      changed to any of the previous 10 passwords."
   "tags":      [ "Password History", "Access Management", ... ]
}
```

Listing 6.1: Control with Meta Information

Topics and *subtopics* group together related control statements, for example, “Password Management” or, more specifically, “Password History”. It is noted that *subtopics* can also be formulated in a general manner, like “General Principles”. The title is basically just a shorter version of the actual statement. *Tags* are, as the name indicates, a collection of related key words.

Table 6.1 gives the average number of words per meta information field in order to get an idea by how much text we extend the control statements.

Meta Information Field	∅ Words
Topic	2.54
Subtopic	2.37
Title	2.69
Tags	8.53
All Fields	16.13
Ground Truth Items	74.30
Control Statements	21.55

Table 6.1.: Meta Information Statistics

These fields often contain overlapping terms and we assume that they do not contain any stopwords. Topic, subtopic and title consists of between 2 and 3 words in average. Tags, on

the other hand, are approximately three times bigger with an average of 8.53 words. So, the sum of the four meta fields is equal to 16.13 words. Together with the average word count of 21.55 per control statements, the final matching strings result in a size of 37.68 words which is roughly the half of the size of the ground truth input controls.

To extend control statements by meta information we use simply string concatenation. Therefore, we write a script which firstly parses the database dump in JSON format that contains all control statements and their meta information. After, it combines statements and the fields topic, subtopic, title and tags.

Figure 6.4 shows our recommender system that matches an input control against control statements that are now extended by tags. Statement 1 and 2 increase their similarities because “Password History, Access Management” strengthens the linkage regarding key word “Password”. The similarity of statement 879, however, decreases because “Data Lifecycle” is an unrelated key term.

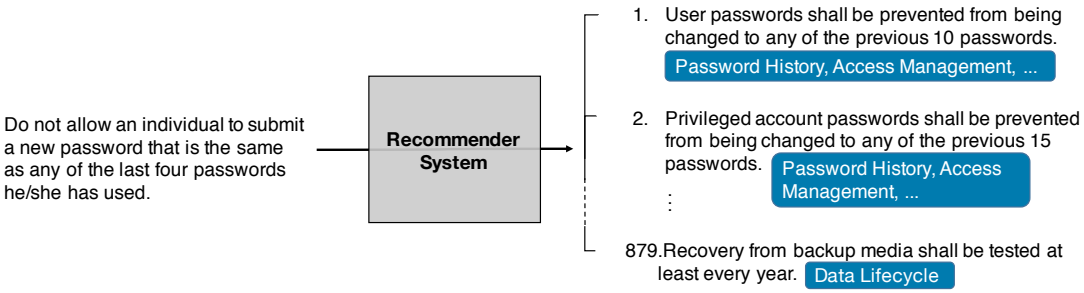


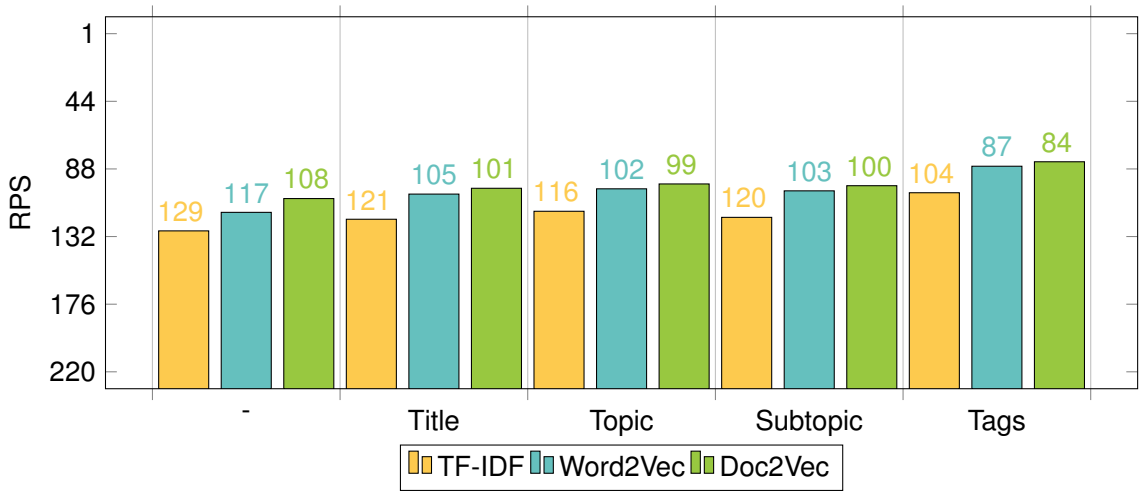
Figure 6.4.: Recommender System with Meta Information

We create 7 Matchers per text similarity approach: 4 Matchers separately add meta information fields to the control statements (CS) - CS+Title, CS+Topic, CS+Subtopic and CS+Tags - and another 3 successively add these meta fields - CS+Title+Topic, CS+Title+Topic+Subtopic and CS+Title+Topic+Subtopic+Tags.

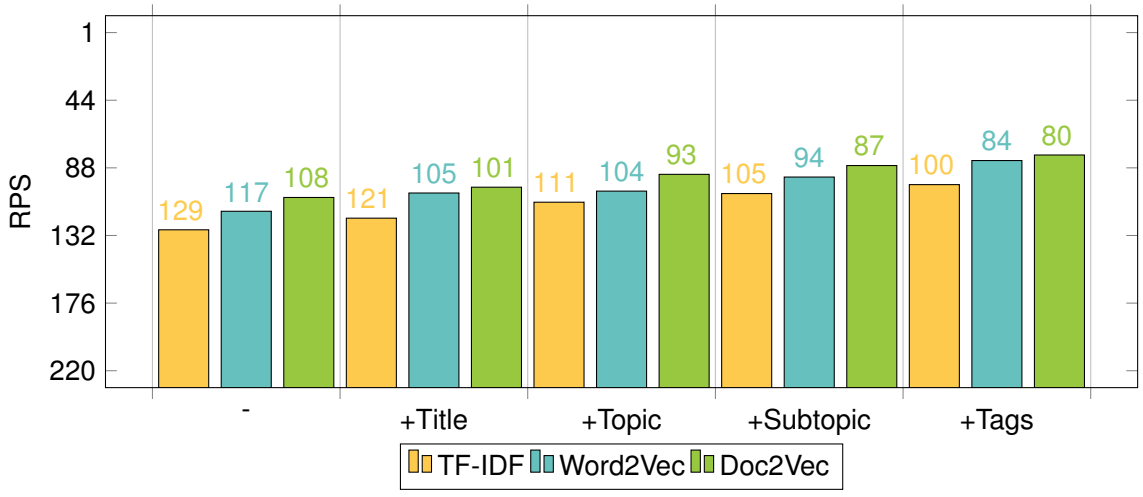
Results

The results show that extending control statements by meta information has a great, positive impact on the matching results, as illustrated in Figure 6.5. The trend is similar for all considered text similarity approaches. The RPS of TF-IDF is increased from 129 to 100 (+29 positions), Word2Vec from 117 to 84 (+33) and Doc2Vec from 108 to 80 (+28).

The separated addition of meta information in Figure 6.5(a) shows that the RPS per meta field is proportional to the average number of words. CS+Title is a minor exception and the reason might be the high similarity of the title with the control statement itself which result in less new matching pairs than other fields create. CS+Tags leads clearly to the best results for TF-IDF (+25), Word2Vec (+30) as wells as Doc2Vec (+24).



(a) Separate Addition



(b) Combined Addition

Figure 6.5.: Control Statements with Meta Information

The combined addition of meta information in Figure 6.5(b) further improves the results and confirms that the RPS increases the more meta information you use. The sum of all meta fields (CS+Title+Topic+Subtopic+Tags) leads to the best results for TF-IDF (+20), Word2Vec (+33) and Doc2Vec (+28).

It must be noted, of course, that the use of meta information reaches its limits: unrelated or general key words as well as terms that are not sufficiently disjunct to other control statements decreases the RPS. With the meta information provided for this thesis, however, we cannot give any negative, real-life examples.

So, in the next iterations, we use control statements that are extended by all available meta information fields: topic, subtopic, title and tags.

6.1.3. Corpora Analysis

This third iteration compares various corpora that differ in size as well as quality (in terms of domain-specific vocabulary) and make a statement about the impact of these characteristics.

Preparation

Due to memory limitations and long computing times during the creation of the Matchers, we aim to create corpora with a maximum size of 400 MB. Although this size is relatively small (other examples from the internet often uses corpora of 5 GB), it is still a very time-consuming task to find sufficient sources. With regard to the research questions to be answered, we decide to use Wikipedia articles as well as regulatory texts from the internet.

Table 6.2 shows the corpora that are taken into account in our analysis.

Name	Description	On-Topic	Size
CTRLS	Control statements	yes	0.1 MB
REG	On-Topic documents	yes	200 MB
WIKI_TOPIC	On-Topic Wikipedia articles	yes	200 MB
WIKI_OTHER	Off-topic Wikipedia articles	no	200 MB
WIKI	WIKI_TOPIC + WIKI_OTHER	mixed	400 MB
TOTAL	REG + WIKI_TOPIC	yes	400 MB

Table 6.2.: Corpora

Wikipedia is a great source for NLP tasks, especially because it provides large and clean texts that you can easily get from a publicly available XML dump[1]. We extend an existing Python script[2] that extracts plain text from Wikipedia and add a function to filter articles by categories. So in order to create a domain-specific corpus we manually pick around 2500 key words which are closely related with the control statements topics (WIKI_TOPIC). We also build another corpus with the non-related fields biology, history, chemistry and geography (WIKI_OTHER). Both files have a size of 200 MB, in total 400 MB (WIKI).

Regulatory texts contains, unlike Wikipedia articles, similar legal formulations like input controls we might find in company policies or regulatory documents. *Alyne* collected 200 MB of plain text (REG). It must be noted that most of the source documents are PDF files which need to be parsed and cleaned. So, this corpus is less clean than the one created from Wikipedia.

As a baseline measure, we create a corpus that consists exclusively of control statements with a very small size of 0.1 MB (CTRLS). Because of the small size, we use 50 epochs instead of 5 for this corpus.

[1] `https://dumps.wikimedia.org/backup-index.html`, last accessed in February 2018
[2] `https://github.com/attardi/wikiextractor`, last accessed in February 2018

Results

The results show that the quality is at least as important as the size of the corpus. The on-topic corpus REG performs best: The RP-Score of Word2Vec is slightly increased from 84 to 82 (+2) and Doc2Vec from 80 to 79 (+1) compared to the second-best corpus WIKI_TOPIC which we have already used in the previous interations. Figure 6.6 illustrates the performance of all corpora.

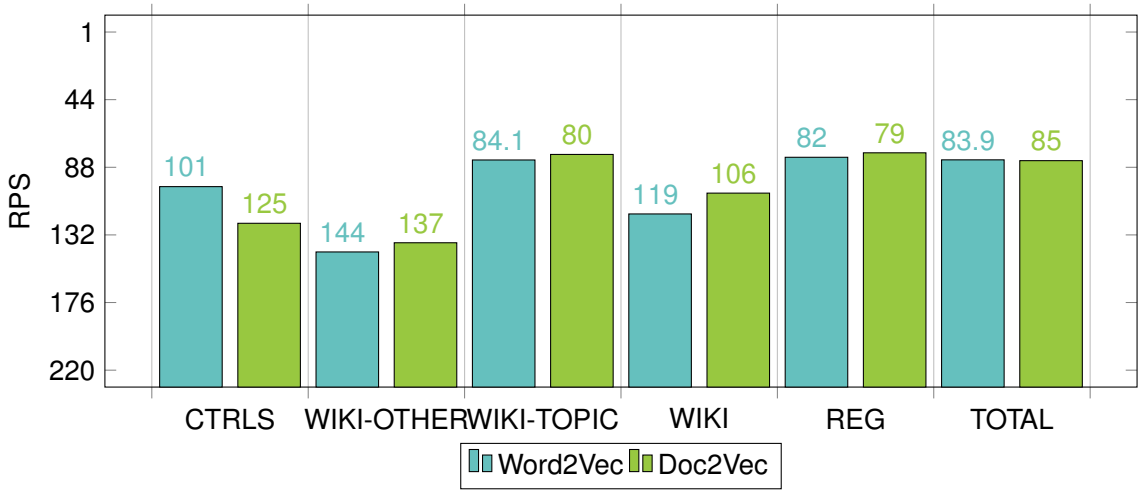


Figure 6.6.: Corpora Analysis

We firstly focus on corpus quality. Let us consider the unrelated WIKI_OTHER (200 MB), the related WIKI_TOPIC (200 MB) and their combination WIKI (400 MB). As a result, WIKI_OTHER performs poorly with a RPS of 144/128 (Word2Vec/Doc2Vec), WIKI_TOPIC performs best with 90/81, and the RPS of WIKI with 118/103 are located in between. It has already been obvious that on-topic corpora build better word vectors. The results emphasises, however, the great deterioration in the results. Even WIKI which contains the best corpus WIKI_TOPIC decreases its RPS by -28/-22 positions because WIKI_OTHER influences the creation of word vectors in a negative way. Here, quality wins compared to quantity.

The positive impact of domain-specific vocabulary is also noticeable when we take a closer look at WIKI_TOPIC (200 MB) and REG (200 MB). The RPS are nearly identical with slight advantages for REG, although it is considered as less clean and more noisy than WIKI_TOPIC.

To evaluate the importance of the corpus size, we consider on-topic corpora of three different sizes: CTRLS (0.1 MB), REG (200 MB) and TOTAL (400 MB). The small CTRLS performs poorly with a RPS of 101/125, the big TOTAL performs better with 83.9/85 but the medium REG performs best with 82/79. This outcome confirms the well-known thumb rule that bigger corpora achieve better results, but also shows the limit of this rule and demonstrates again the importance of corpus quality. Although both corpora are on-topic, a combination does not automatically lead to an improved matching result. Regarding corpus CTRLS, Doc2Vec seems to work poorly with very small corpora.

In the next iteration the machine learning based approaches continue with the corpus REG.

6.1.4. Context Information

This fourth and last iteration examines whether chapter or paragraph context help to improve the matching results.

The idea is similar to the addition of meta information from Section 6.1.2. Instead of adding meta information to control statements, we here extend input controls by context information such as section titles. However, the objective remains the same: we aim to create new matching pairs which boost up related controls statements.

Preparation

Similar controls within company policies or regulatory documents are usually grouped together under a fitting heading which can contain important key words. These might support the algorithm by creating new matching pairs.

To extend input controls by context information we again use simply string concatenation. Together with two colleagues from *Alyne*, we collect and integrate the missing section and paragraph title, if any, for each ground truth item. In production mode, this job could be done by an intelligent parser that can recognize relevant headings.

Table 6.3 shows the average number of words per title which we plan to add to the corresponding input controls. The values vary between 3.35 and 13.26 words in average.

Regulatoy Document	#Paragraphs	∅ Words/Title
BDSG	24	13.26
COBIT 5	191	10.78
COSO	16	4.63
Finish Personal Data Act	29	11.73
ISO 22301	37	7.11
ISO 27001	133	9.74
ISO 31000	46	6.2
MAS-TRMG	268	7.11
NIST	95	3.35
PCI DSS	161	13.63
All Ground Truths	1000	9.45

Table 6.3.: Ground Truths with Title

Figure 6.7 shows our recommender system that now extends input controls by their section titles, for example “Implement strong access control measure”. The resulting string is then

matched against all control statements. Statement 1 and 2 increase their similarity because of the newly created matching pair “access”. The similarity of statement 879 further decreases because the added context information only contains unrelated words.

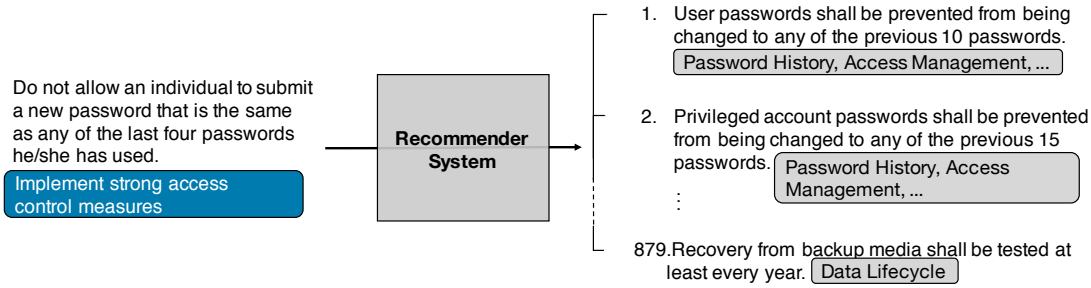


Figure 6.7.: Input Controls with Context Information

For this iteration, we create 1 Matcher per text similarity approach that matches input controls, which are now extended by context information, against all control statements.

Results

The results show that extending input controls by context information has a great, positive impact on the matching results for all considered text similarity approaches, as illustrated in Figure 6.8. The RP-Score of TF-IDF is increased from 100 to 86 (+24 positions), Word2Vec from 82 to 77 (+5) and Doc2Vec from 79 to 71 (+8).

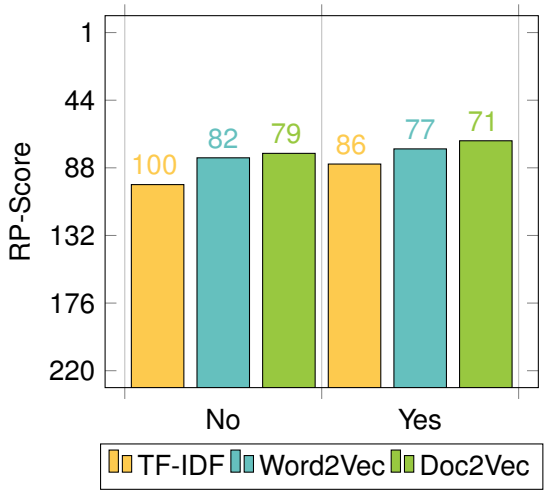


Figure 6.8.: Input Controls with Context Information

In this setup, TF-IDF benefits from context information considerably more than Word2Vec or Doc2Vec does. This fact allows us to conclude that the titles in our ground truths mainly create direct matching pairs between exactly same words instead of indirect matching pairs through, for example, synonyms.

6.2. Overall Evaluation

After improving the matching results by the previously performed individual evaluations, we now focus on an overall evaluation with regard to

- the performance of the text similarity approaches (research question 2) and,
- finally, the semantic matching problem that *Alyne* faces (research question 1).

In simple words, both sections/questions basically deal with the same question: How reliable do text similarity approaches (or the recommender system) link semantically related text passages?

6.2.1. Text Similarity Approach Analysis

Figure 6.9 summarizes the best performing Matchers per text similarity approach by comparing the RPS values of the initial and final matching results.

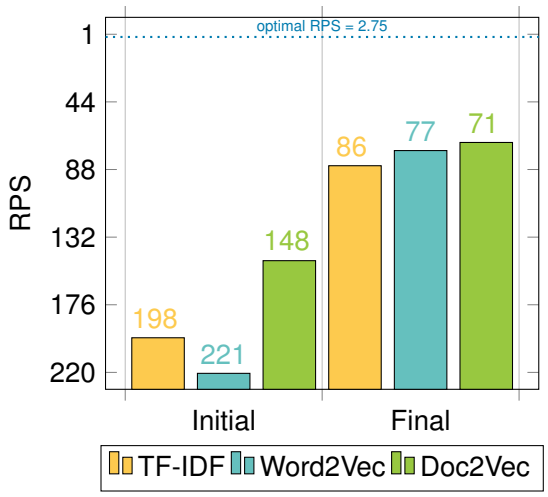


Figure 6.9.: Final Evaluation

While the text similarity approaches differs materially in the beginning, they have improved a lot and the final results are more balanced in the end. Especially preprocessing has boosted up TF-IDF as well as Word2Vec and shorten the gap to Doc2Vec. Other strategies such as adding meta or context information improves the performance of the approaches in a similar manner.

Regarding the final RPS values, the machine learning approaches perform better than TF-IDF does, in particular the Doc2Vec implementation with an average ranking position of 71. Word2Vec has a final score of 77 (-6 positions) and TF-IDF even 86 (-15). Despite the fact that Word2Vec and Doc2Vec can handle with synonyms (to some degree) and TF-IDF cannot, the simple and efficient TF-IDF still works very well.

Figure 6.10 examines the final performance from a different perspective. The RPS is a simple and intuitive evaluation measure but limited in its explanatory power. In order get a deeper insight into the results, we modify the previously used plot type by adding a new dimension that shows the percentage of ground truth items per ranking in intervals of 50 positions - 1+ means in between rank 1 and 49. So, in an ideal world, all items would be ranked with the optimal RPS, which means that 1+ is equal to 100 for all approaches in this example. The results show, however, rankings appearing in each interval.

The first conclusion that can be stated is that the rankings of TF-IDF, Word2Vec as wells as Doc2Vec are very similarly distributed. More than half of the rankings are located in 1+ (TF-IDF 54.2%, Word2Vec 50.0%, Doc2Vec 53.7%), and (about) more than two third in 1+ together with 50+, which means positions in between 1 and 99 (TF-IDF 66.3%, Word2Vec 63.8%, Doc2Vec 67.3%). The percentage of ground truth items declines with increasing intervals more or less consistently. The lower part between 450+ and 850+, from position 145 to 879, only contains less than 10% of all ranking results (TF-IDF 10.1%, Word2Vec 8.7%, Doc2Vec 7.7%).

Although the rankings of all considered approaches are similarly distributed, we still can see differences between TF-IDF and the machine learning approaches, and state a second conclusion. While TF-IDF results in many positive matchings and also many comparatively bad matchings, Word2Vec and Doc2Vec provide better balanced results. This behaviour can be explained by TF-IDF's accurate matchings and word embeddings' understanding for synonyms. Despite the fact that Doc2Vec performs significantly better than TF-IDF does (difference of 15 positions), TF-IDF ranks more ground truth items into interval 1+.

6.2.2. STM Problem Analysis

Based on the performed text similarity approach analysis, we now investigate whether and to what extent our recommender system solves the STM problem of company policies and regulatory documents.

First, we discuss the outliers from Figure 6.10 to estimate further potential for improvement and to identify common problems. Second, the level of maturity of the recommender system for the productive use is assessed.

Therefore, four experts examine the 209 worst ranked ground truth items (Expert 1: 60, Expert 2: 60, Expert 3: 59, Expert 4: 30) independently of one another using the feedback functionality of *Citadel* (see Section 5.2.2).

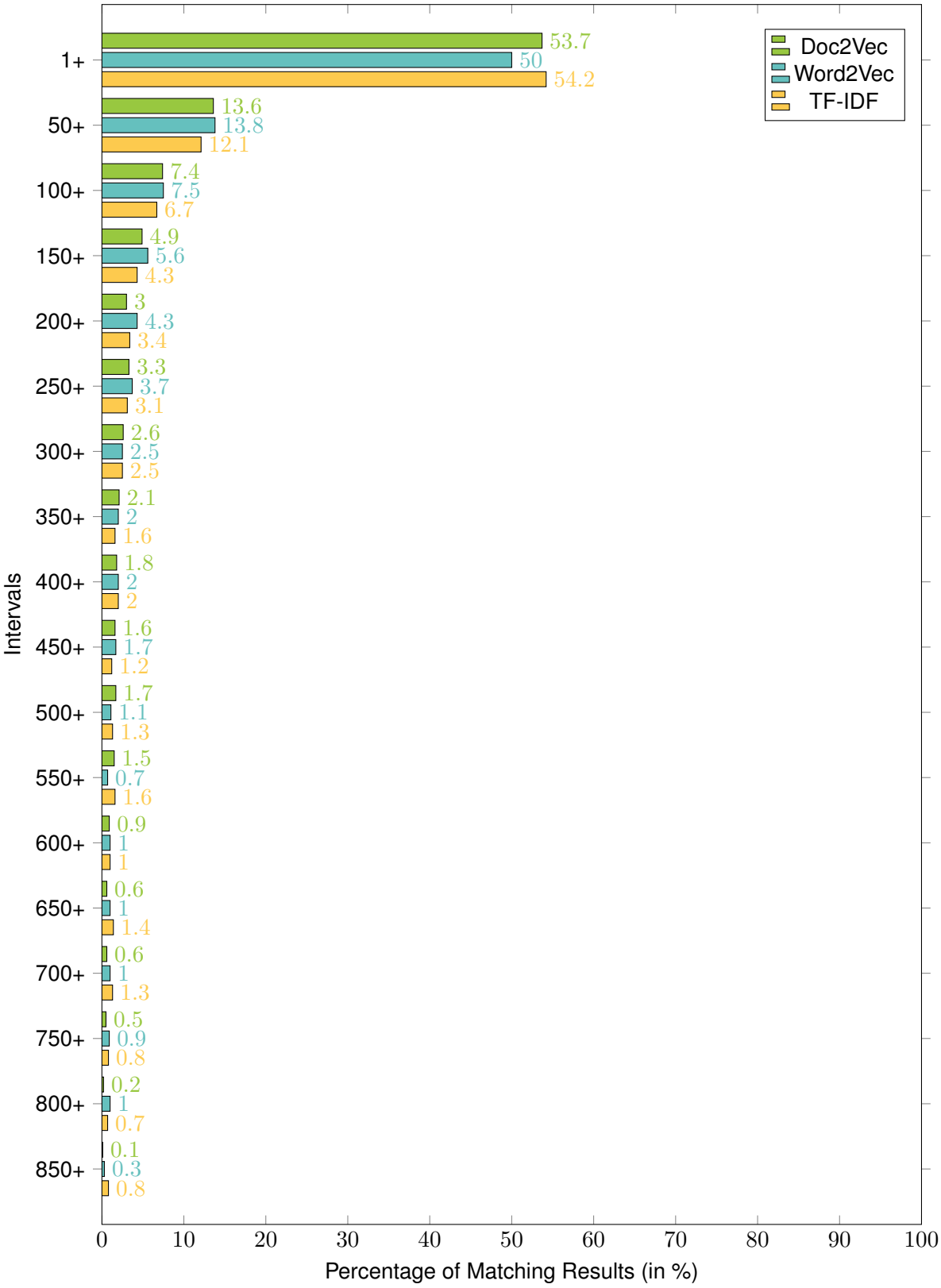


Figure 6.10.: Distribution of Matching Results

For each of the 209 outliers, the experts focus on four questions:

- Are there any reoccuring patterns?
- How similar is the input control with the **expected** output control statements?
  In ideal case, the input and expected output are similar enough to successfully match them. **In cases of outliers**, however, the output is not correctly matched so that we assume dissimilar vocabulary which is hard for the algorithm to "understand".
  Idea: No similarity indicates correct functioning of the algorithm.
- How similar is the input control with the **actual** output control statements (top 10)?
  In ideal case as well as cases of outliers, we assume the top ranked outputs to be similar.
  Idea: High similarity indicates correct functioning of the algorithm.
- Is the recommender system ready for productive use?

Detected Patterns

During their evaluation the experts detect the following patterns:

- abbreviations,
- short input,
- wrong focus,
- missing synonyms,
- specific input control and general control statement and
- general input control and specific control statement.

Abbreviations are avoided as far as possible by *Alyne*'s control statements to prevent ambiguity. Other sources, however, frequently use acronyms which cannot be mapped consequently (for example, SAN and Storage Area Network). A solution might be to extend the meta information of a control statement by abbreviations or to preprocess input controls by replacing short forms through full forms using the document's list of abbreviation, if available, or any online service.

Short input controls are often used due to their good comprehensibility but misses relevant context information. This problem can be solved by extracting section titles, as we showed in Section 6.1.4, but also by taking into account the previous and next paragraph, or even the document's table of contents. An extreme case for short inputs is an enumeration which consists of one sentence, goes over several lines and includes multiple paragraphs with own numbers (§1,§2 etc.) which should be probably matched individually.

A wrong focus can be already corrected by the solution ideas for abbreviations or short input.

We additionally observe the need of extraction and tranformation of keyphrases. For example, both “IT assets” and “information assets” needs to be tranformed from two words to one single token before tokenizing so that they become more distinctive from each other for the algorithm.

While missing synonyms can be “easily” improved by adding context information to input or meta information to control statements, the imbalance in depth of content between input control and control statement represents a far greater challenge. A possible solution is to match, instead of the control statement, already (manually) mapped controls which might have same depth of content.

Simlarity between Input Control and Expected Control Statements

We now compare the similarity between input control and the expected control statements. Since we consider the worst rankings, we expect a very low similarity, because otherwise, the algorithm should work and match in a correct manner.

Therefore, the experts classify the vocabulary of input and output into exactly one of four categories: different vocabulary, same topic of vocabulary, synonyms or same key words.

Figure 6.11 shows the experts’ assessment:

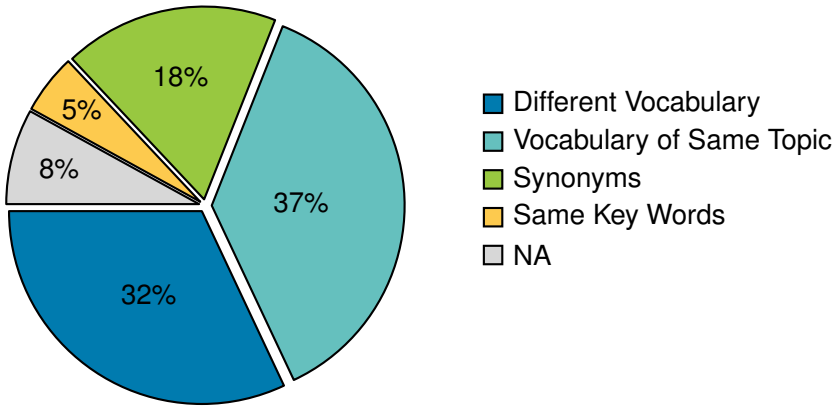


Figure 6.11.: Similarity between Input Controls and Expected Control Statements

The results confirm our expectations.

Almost 70% of all matching results have different or only topic-related vocabulary which is hard to learn by the algorithm and consequently difficult to link. Adding more information to input or output can support the algorithm.

Synonyms, however, should be captured and thus achieve higher rankings. The percentage of 18% for segment “Synonyms” indicates potential for parameter optimization.

Only 5% of the worst rankings contains key words in both input and expected output which we can generally explain and solve with the introduced solutions to the detected patterns.

Simlarity between Input Control and Actual Control Statements

In cases where the expected controls are not that related, we at least assume that the actually matched (top 10) results are similar in terms of keywords, synonyms or topic.

Therefore, for each ground truth item we compare the input control with the 10 highest ranked control statements and determine the number of cases where both use similar vocabulary.

Figure 6.12 shows the experts' assessment:

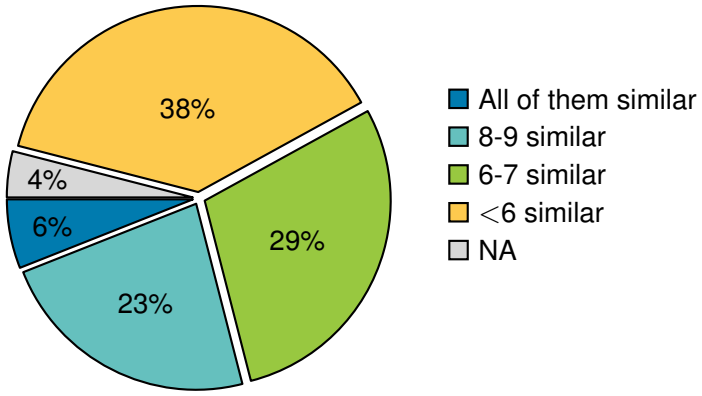


Figure 6.12.: Similarity between Input Control and Actual Top 10 Control Statements - similar in terms of Keywords, Synonyms or Topic

The results do not confirm our expectations.

In only 29% of the cases, the top 10 results contains 8 or more similar control statements, just as much as in between 6 and 7. In even 38% of the cases, less than 6 control statements are related.

In this regard, it can be concluded that these input controls might provide poor information. With an average number of words of 51.41, the input control of the ground truth items in segment "<6 similar" are considerably shorter compared to the total average of 83.75 words. (74.3 from Table 1.2 in Section 1.3.2 + 9.45 from Table 6.3 in Section 6.1.4).

Level of Maturity of the Recommender System

Finally, we estimate the level of maturity of the recommender system for the productive use.

As a reminder, our approach supports regulatory experts, during their mapping process of company policies and regulatory documents, by recommending related control statements. We aim to provide a list of all control statements which is sorted according to the similarity with the input control while related ones appear at the very top. The traditional method is a completely manual approach.

We define four levels of maturity for the recommender system:

1. Expert works (almost) exclusively with recommender system
   Effect: Significant time savings

2. Expert works with recommender system (50%) but also traditional method (50%)
   Effect: Medium time savings

3. Expert uses recommender system only as cross-check and reassurance
   Effect: Improved data quality

4. No productive applicability yet

The expert interview show that the recommender system reaches level 3 with a tendency to level 2, which means that the algorithm can already contribute actively to a higher data quality. A simple example might be a cross-check during a manual mapping to ensure no relevant control statements have been overlooked.

To reach level 2, it is necessary to further decrease the number of outliers. This task can be achieved by developing countermeasures for the identified patterns - some solution ideas are already introduced in this chapter. It is noted that an intuitive user interface, which allows the expert to consume the recommendations, is a key success factor.

In summary, text similarity approaches solve (to a certain degree) the STM problem of regulatory documents and company policies.

7. Conclusion and Future Work

This thesis presented that text similarity approaches successfully support regulatory experts to match semantically related controls from company policies and regulatory documents. It is an instance of the semantic text matching (STM) problem.

We developed a recommender system that finds, for a given control, related generalized control statements from a predefined collection. The system implements the unsupervised text similarity algorithms Word2Vec and Doc2Vec, as well as the classical TF-IDF approach as a baseline. For evaluation purposes, we build up an user interface that allows to manage differently configured instances of the recommender system, to visualize matching results and to enable structured, qualitative feedback provided by regulatory experts.

With the aim to enhance the performance of the algorithms, four different techniques were quantitatively assessed including preprocessing technologies, addition of meta information, corpora analysis and addition of paragraph context. It showed that all these techniques lead to improved matching results for all text similarity approaches. We state that a domain-specific corpus has a decisive influence on the matching quality. While the initial evaluation without any optimization showed better results for TF-IDF compared to Word2Vec, the final evaluation presents contrary results. Doc2Vec, however, achieved generally the best results.

A qualitative evaluation of bad matching results was performed by regulatory experts who identified five patterns: abbreviations, short input, missing synonyms, wrong focus, and imbalance in depth of content between input and expected output. We also observed that the input and output of bad results often do not share similar vocabulary which is obviously difficult to be matched by an algorithm. The expert interview showed that the recommender system in its current version is already able to actively support experts but the number of outliers is still high.

Through the analysis using *Citadel*, we also identified a number of cases where the manual initial mapping had errors. We also identified cases where the issues of a standard had applied incorrect matchings within the standard. This demonstrates a superior capability of *Citadel* over a manual approach in many cases.

In future work, we need to further improve the matching quality and thus reduce outliers by developing countermeasures for the detected patterns. Examples are parameter optimization, replacing abbreviations through its full form, adding more context information to controls or transforming keyphrases to single tokens. A more advanced method is to match, instead of the control statements, already (manually) mapped controls. It is also feasible to make use not of one but more similarity approaches, for example, TF-IDF and Doc2Vec in combination and merge their results.

An intelligently designed user interface for experts to consume the recommendations and find relevant control statements might also compensate deficiencies of the algorithm, for example, by highlighting key words and synonyms, or providing filter options for the current control topic.

Bibliography

[Dai et al., 2015] Dai, A. M., Olah, C., and Le, Q. V. (2015). Document embedding with paragraph vectors. In *NIPS Deep Learning Workshop*.

[Duan and Xu, 2016] Duan, L. and Xu, T. (2016). A short text similarity algorithm for finding similar police 110 incidents. In *2016 7th International Conference on Cloud Computing and Big Data (CCBD)*, pages 260–264.

[Gomaa and Fahmy, 2013] Gomaa, W. and Fahmy, A. (2013). A survey of text similarity approaches. *International Journal of Computer Applications*, 68.

[Gurusamy and Kannan, 2014] Gurusamy, V. and Kannan, S. (2014). Preprocessing techniques for text mining.

[Kusner et al., 2015] Kusner, M. J., Sun, Y., Kolkin, N. I., and Weinberger, K. Q. (2015). From word embeddings to document distances. In *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, pages 957–966. JMLR.org.

[Landthaler, 2017] Landthaler, J.; Waltl, B. H. D. B. D. S. C. G. T. M. F. (2017). Improving thesauri using word embeddings and a novel intersection method. *Proc. of 2nd Workshop Workshop on Automated Semantic Analysis of Information in Legal Texts (ASAIL'17)*.

[Landthaler, 2018] Landthaler, J.; Scepankova, E. G. I. L. H. M. F. (2018). Semantic text matching of contract clauses and legal comments in tenancy law. *IRIS: Internationales Rechtsinformatik Symposium*.

[Lau and Baldwin, 2016] Lau, J. H. and Baldwin, T. (2016). An empirical evaluation of doc2vec with practical insights into document embedding generation. *CoRR*, abs/1607.05368.

[Le and Mikolov, 2014] Le, Q. V. and Mikolov, T. (2014). Distributed representations of sentences and documents. *CoRR*, abs/1405.4053.

[Mikolov et al., 2013a] Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013a). Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781.

[Mikolov et al., 2013b] Mikolov, T., Sutskever, I., Chen, K., Corrado, G., and Dean, J. (2013b). Distributed representations of words and phrases and their compositionality. *CoRR*, abs/1310.4546.

[Naderi and Hirst, 2015] Naderi, N. and Hirst, G. (2015). Argumentation mining in parliamentary discourse. In *Principles and Practice of Multi-Agent Systems - International Workshops: IWEC 2014, Gold Coast, QLD, Australia, December 1-5, 2014, and CMNA XV and IWEC 2015, Bertinoro, Italy, October 26, 2015, Revised Selected Papers*, pages 16–25.

[Rinott et al., 2015] Rinott, R., Dankin, L., Perez, C. A., Khapra, M. M., Aharoni, E., and Slonim, N. (2015). Show me your evidence - an automatic method for context dependent evidence detection. In Màrquez, L., Callison-Burch, C., Su, J., Pighin, D., and Marton, Y., editors, *EMNLP*, pages 440–450. The Association for Computational Linguistics.

[Salton and Buckley, 1988] Salton, G. and Buckley, C. (1988). Term-weighting approaches in automatic text retrieval. *Information Processing Management*, 24(5):513 – 523.

[Salton et al., 1975] Salton, G., Wong, A., and Yang, C. S. (1975). A vector space model for automatic indexing. *Commun. ACM*, 18(11):613–620.

[Vijayarani and Ilamathi, 2015] Vijayarani, D. S. and Ilamathi, J. (2015). Preprocessing techniques for text mining - an overview. In *International Journal of Computer Science and Communication Networks*, pages 7–16.

[Waltl, 2015] Waltl, B.; Zec, M. M. F. (2015). Lexia: A data science environment for legal texts. *Jurix: International Conference on Legal Knowledge and Information Systems*.

List of Figures

1.1. Implicit References between Regulatory Documents and Company Policies . . . 1
1.2. Explicit References between Regulatory Documents and Company Policies . . 3
1.3. Work Routine of Regulatory Experts . . . 3
1.4. Recommender System using a Text Similarity Approach . . . 4
1.5. Iterative Research Approach . . . 6

2.1. Illustration of STM Problem, based on [Landthaler, 2018] . . . 10

3.1. Preprocessing Pipeline, based on [Vijayarani and Ilamathi, 2015] . . . 12
3.2. Word2Vec models, based on [Mikolov et al., 2013a, Le and Mikolov, 2014] . . . 16
3.3. Doc2Vec models, based on [Le and Mikolov, 2014] . . . 18

4.1. System Components . . . 20
4.2. Configuration Class Diagram . . . 21
4.3. Example Configuration File . . . 21
4.4. Matcher Class Diagram . . . 22
4.5. Data Sets Class Diagram . . . 23

5.1. Deployment Diagram of Citadel . . . 25
5.2. Run and Evaluation Management . . . 26
5.3. Dashboard . . . 27
5.4. Demo . . . 28
5.5. Evaluation Details . . . 29
5.6. Details of Ground Truth Item . . . 29
5.7. Feedback Loop . . . 30

6.1. Initial Evaluation . . . 31
6.2. Cleaning, Stopword Removal and Stemming . . . 33
6.3. PoS Tagging . . . 34
6.4. Recommender System with Meta Information . . . 36
6.5. Control Statements with Meta Information . . . 37
6.6. Corpora Analysis . . . 39
6.7. Input Controls with Context Information . . . 41
6.8. Input Controls with Context Information . . . 41
6.9. Final Evaluation . . . 42
6.10. Distribution of Matching Results . . . 44
6.11. Similarity between Input Controls and Expected Control Statements . . . 46
6.12. Similarity between Input Control and Actual Top 10 Control Statements - similar in terms of Keywords, Synonyms or Topic . . . 47

List of Tables

1.1. Example controls from regulatory documents and company policies 2
1.2. Ground Truth 7
1.3. Matching Result Example 8

3.1. Exemplary Application of Preprocessing Technologies 13

6.1. Meta Information Statistics 35
6.2. Corpora 38
6.3. Ground Truths with Title 40

A.1. Stemming, Cleaning, Stopwords Removal 56
A.2. PoS Tagging 57
A.3. Meta Information 58
A.4. Corpora Analysis 59
A.5. Context Information 59

Appendix

A. Result Tables

Preprocessing

Algorithm	Stemming	Cleaning	Stopwords	FRP	RPS	LRP	%RPS
TFIDF	No	No	No	89.68	198.05	339.04	77.56
	No	No	Yes	91.16	194.51	316.96	77.96
	No	Yes	No	71.47	164.74	294.31	81.35
	No	Yes	Yes	73.06	163.15	275.20	81.53
	Yes	No	No	55.98	136.90	249.00	84.52
	Yes	No	Yes	53.14	129.98	228.73	85.31
	Yes	Yes	No	53.63	132.46	242.94	85.03
	Yes	**Yes**	**Yes**	**52.77**	**129.00**	**227.25**	**85.42**
Word2Vec	No	No	No	113.58	221.12	355.55	74.93
	No	No	Yes	67.99	143.33	249.49	83.79
	No	Yes	No	95.66	193.92	323.03	78.03
	No	**Yes**	**Yes**	**50.90**	**116.94**	**212.32**	**86.79**
	Yes	No	No	100.35	207.28	343.21	76.51
	Yes	No	Yes	61.71	139.15	248.72	84.27
	Yes	Yes	No	104.79	210.08	343.99	76.19
	Yes	Yes	Yes	58.90	133.88	237.84	84.87
Doc2Vec	No	No	No	70.51	147.94	257.91	83.26
	No	No	Yes	52.36	120.47	222.54	86.39
	No	Yes	No	57.54	129.93	233.00	85.32
	No	**Yes**	**Yes**	**45.56**	**108.00**	**200.14**	**87.81**
	Yes	No	No	62.67	136.72	244.17	84.54
	Yes	No	Yes	48.47	112.72	210.48	87.28
	Yes	Yes	No	55.78	129.18	235.82	85.40
	Yes	Yes	Yes	46.36	108.38	204.64	87.77

Table A.1.: Stemming, Cleaning, Stopwords Removal

Algorithm	PoS Tags	FRP	RPS	LRP	%RPS
TFIDF	-	52.77	129.00	227.25	85.42
	N	76.88	160.51	259.34	81.83
	NV	63.01	140.11	237.57	84.16
	NA	64.50	143.98	242.23	83.72
	NVA	**53.27**	**128.77**	**226.30**	**85.45**
Word2Vec	-	50.90	116.94	212.32	86.79
	N	64.50	133.25	233.42	84.94
	NV	52.92	119.18	215.46	86.54
	NA	55.13	121.71	218.66	86.25
	NVA	**48.93**	**114.06**	**208.21**	**87.12**
Doc2Vec	-	45.56	108.00	200.14	87.81
	N	58.91	123.42	222.80	86.06
	NV	48.77	110.58	204.49	87.52
	NA	49.69	113.94	211.75	87.14
	NVA	**45.09**	**106.89**	**199.24**	**87.94**

Table A.2.: PoS Tagging

Editorial Help

Algorithm	Meta-Information	FRP	RPS	LRP	%RPS
TFIDF	DOC	52.77	129.00	227.25	85.42
	DOC_TITLE	48.56	121.45	217.25	86.28
	DOC_TOPIC	50.57	116.27	200.89	86.87
	DOC_SUBTOPIC	51.84	120.22	207.08	86.42
	DOC_TAGS	48.56	104.28	177.72	88.24
	DOC_TITLE_TOPIC	47.32	111.07	193.34	87.46
	DOC_TITLE_TOPIC_SUBTOPIC	46.84	105.48	179.73	88.10
	DOC_TITLE_TOPIC_SUBTOPIC_TAGS	**47.01**	**99.66**	**167.08**	**88.76**
Word2Vec	DOC	50.90	116.94	212.32	86.79
	DOC_TITLE	44.97	105.11	192.13	88.14
	DOC_TOPIC	48.01	101.73	180.21	88.53
	DOC_SUBTOPIC	45.08	103.06	186.52	88.38
	DOC_TAGS	**41.49**	87.06	151.69	90.20
	DOC_TITLE_TOPIC	48.29	103.94	183.41	88.28
	DOC_TITLE_TOPIC_SUBTOPIC	43.12	94.80	168.10	89.32
	DOC_TITLE_TOPIC_SUBTOPIC_TAGS	42.11	**84.10**	**141.81**	**90.54**
Doc2Vec	DOC	45.68	108.21	201.20	87.79
	DOC_TITLE	41.87	101.36	189.63	88.57
	DOC_TOPIC	43.75	98.55	177.09	88.89
	DOC_SUBTOPIC	42.89	99.66	181.83	88.76
	DOC_TAGS	**38.16**	84.14	151.78	90.53
	DOC_TITLE_TOPIC	39.98	93.04	170.73	89.52
	DOC_TITLE_TOPIC_SUBTOPIC	38.49	87.29	157.71	90.17
	DOC_TITLE_TOPIC_SUBTOPIC_TAGS	38.39	**80.45**	**141.59**	**90.95**

Table A.3.: Meta Information

Corpus

Algorithm	Corpus	FRP	RPS	LRP	%RPS
Word2Vec	CTRLS	55.69	101.39	161.96	88.57
	WIKI_OTHER	82.01	143.77	224.31	83.74
	WIKI_TOPIC	42.11	84.10	141.81	90.54
	WIKI	63.34	119.12	194.47	86.55
	REG	**39.02**	**82.33**	**142.16**	**90.74**
	TOTAL	40.57	83.93	144.39	90.55
Doc2Vec	CTRLS	68.93	125.14	205.36	85.86
	WIKI_OTHER	77.24	137.79	217.78	84.42
	WIKI_TOPIC	38.39	80.45	141.59	90.95
	WIKI	52.78	105.55	178.77	88.09
	REG	**38.48**	**79.38**	**140.41**	**91.07**
	TOTAL	41.20	84.50	148.07	90.49

Table A.4.: Corpora Analysis

Context Information

Algorithm	Context Information	FRP	RPS	LRP	%RPS
TFIDF	No	47.01	99.66	167.08	88.76
	Yes	**41.84**	**86.31**	**145.96**	**90.28**
Word2Vec	No	39.02	82.33	142.16	90.74
	Yes	**36.98**	**76.51**	**132.56**	**91.40**
Doc2Vec	No	38.48	79.38	140.41	91.07
	Yes	**34.15**	**71.21**	**127.23**	**92.00**

Table A.5.: Context Information