UNIVERSITÄT ZU LÜBECK

From the Institute of Information Systems
of the University of Lübeck
Director: Prof. Dr. rer. nat. habil. Ralf Möller

Context is the Key:

Context-aware Corpus Annotation using Subjective Content Descriptions

Dissertation
for Fulfilment of
Requirements
for the Doctoral Degree
of the University of Lübeck

from the Department of Computer Sciences

Submitted by

Felix Kuhr
from Hamburg

Lübeck 2021

First referee	Prof. Dr. rer. nat. habil. Ralf Möller
Second referee	Prof. Dr. Dr. habil. Karl-Heinz Zimmermann
Date of oral examination	March 24, 2022
Approved for printing. Lübeck	

Abstract

Adding interpretative linguistic data to an individual collection of documents is known as corpus annotation. Corpus annotation is widely accepted as one of the key contributions to the benefit of a corpus, since annotations enrich a corpus with additional data that is valuable for different tasks performed on a corpus. Over the last sixty years, different kinds of annotations have been introduced with the intent that the annotations should add a value to a corpus for tasks such as Information Retrieval (IR). Generally, interpretative linguistic data is the product of text understanding and a specific task. Thus, interpretative linguistic data is not objective, e.g., an annotation in form of an explanation about an action in a text highly depends on the annotator the context represented by the individual collection of documents and a corpus. So, different valid explanations might exist for the same action in a document. Since manually adding data to documents is time-consuming, in recent years more and more use has been made of semi-automatic annotation systems. However, available annotation systems are document-oriented and ignore the unique context of a document represented by an individual collection of documents in a corpus. The problem studied in this dissertation is the problem of *estimating context-aware Subjective Content Descriptions (SCDs) for documents in a corpus adding a value for corpus-specific tasks.* We introduce a new kind of corpus annotations denoted as SCDs and use the idea of topic modeling approach Latent Dirichlet Allocation (LDA) to solve the problem by estimating a context-aware relevance value for SCDs based on the corpus-driven topics of documents as well as SCDs associated to documents. The contributions of this dissertation are categorized into two parts: In the first part, we present the foundation of SCDs followed by a context-specific corpus enrichment approach for new documents. It follows an approach to estimate a document-specific relevance value for SCDs which is used in a corpus-driven enrichment process associating weakly annotated documents with SCDs that are associated with similar documents in the same corpus. Since SCDs might be encoded directly into the text, we present an approach to identify hidden SCDs within the text of documents. Finally, we present a context-aware adaptation approach to automatically estimate SCDs for documents in a new corpus which are not associated with SCDs. In the second part, we focus on results regarding topic modeling techniques in the domain of SCDs and provide techniques to compare document-topic probability distributions with each other resulting from different topic models and we present results for enhancing relational topic models with named entity induced links between documents in a corpus.

Kurzfassung

Das Hinzufügen interpretativer linguistischer Daten zu Dokumenten in einem Korpus wird als Korpusannotation bezeichnet. Das Annotieren eines Korpus wird als einer der wichtigsten Beiträge zum Nutzen eines Korpus verstanden, da Annotationen den Korpus mit zusätzlichen Daten anreichern, die für bestimmte Aufgaben nützlich sein können die mit dem Korpus durchgeführt werden. In den letzten sechzig Jahren wurden verschiedene Arten von Annotationen eingeführt. Allen Annotationen ist gemeinsam, dass sie einen Mehrwert für Aufgaben darstellen sollen wie z.B. der Informationsgewinnung aus Dokumenten eines Korpus. Im Allgemeinen werden interpretative Annotationen aus dem Textverständnisses gewonnen und sind daher nicht objektiv. Ob eine Annotation eine Erklärung über eine im Text vorkommende Handlung darstellt, hängt vom jeweiligen Annotator ab, sodass es unterschiedlich geeignete Annotationen geben kann. Dadurch, dass das manuelle Hinzufügen interpretativer Daten zu Dokumenten zeitaufwendig ist, wurden in den letzten Jahren zunehmend semi-automatische Annotationssysteme eingeführt. Die verfügbaren Annotationssysteme sind dokumentenorientiert und ignorieren den Kontext eines Dokuments, der durch die jeweilige Sammlung der Dokumenten repräsentiert wird. Das in dieser Dissertation untersuchte Problem ist daher das Problem der *Bestimmung von kontextbewussten subjektiven Inhaltsbeschreibungen (SCDs) für Dokumente in einem Korpus, sodass die Inhaltsbeschreibungen einen Mehrwert für korpusspezifische Aufgaben bieten.*

Wir kombinieren die Idee der sog. Themenmodellierung mit einer neuen Kategorie von Korpusannotation, die wir als subjektive Inhaltsbeschreibung bezeichnen, mit dem Ziel, das genannte Problem zu lösen und die Performanz von unterschiedlichen Aufgaben der Annotation zu erhöhen. Wir konzentrieren uns auf den Ansatz der Latent Dirichlet Allocation (LDA)-Themenmodellierung. Bei dieser Modellierung werden die Dokumente mittels einer Wahrscheinlichkeitsverteilung über Themen darstellt, um die kontextbezogene Ähnlichkeit zwischen Dokumenten in einem Korpus auf der Grundlage der Themen sowie der Inhaltsbeschreibungen zu identifizieren.

Wir gliedern die Arbeit in die nachfolgenden zwei Teile. Zunächst präsentieren wir Grundlagen der subjektiven Inhaltsbeschreibungen, gefolgt von einem Ansatz zur kontextspezifischen Korpusanreicherung neuer Dokumente. Anschließend präsentieren wir einen Ansatz, um einen dokumentenspezifischen Relevanzwert von SCDs zu errechnen, und wir nutzen diesen Relevanzwert in einem korpusgesteuerten Prozess zur Anreicherung von schwach annotierten Dokumenten mit zusätzlichen Inhaltsbeschreibungen, die mit ähnlichen Dokumenten innerhalb desselben Korpus assoziiert sind. Da Inhaltsbeschrei-

bungen auch direkt im Text kodiert sein können, stellen wir ebenfalls einen Ansatz vor, um kodierte Inhaltsbeschreibungen innerhalb eines Texts zu identifizieren. Schließlich stellen wir einen Ansatz vor, um Inhaltsbeschreibungen für Dokumente, basierend auf den Inhaltsbeschreibungen eines anderen Korpus, automatisch zu generieren. Im zweiten Teil dieser Arbeit konzentrieren wir uns auf Ergebnisse zu Themenmodellierungstechniken im Bereich der Inhaltsbeschreibungen und stellen Techniken zum Vergleich von Themenverteilungen vor, die sich aus verschiedenen Themenmodellen ergeben. Wir präsentieren Ergebnisse zur Erweiterung eines relationalen Themenmodells mit Verknüpfungen zwischen Entitäten die im Text der Dokumente vorkommen.

Acknowledgements

Writing this dissertation is a resume of the last five years being a PhD student at the University of Lübeck. This document combines the work of thousands of hours thinking and discussing about problems, working on solutions for those problems, writing academic papers to publish the own work, and swap ideas with other researchers at conferences.

I remember the time back when I was a master's student working on my thesis and have supported you Mona in evaluating the approaches for academic papers about indirect dependencies between network services. It was a great time. However, I had a lot of difficulty in writing an evaluation for our results in an academic style. Today, I am writing the final section of my dissertation and I am happy about the experiences I have gathered over the last years. So, thank you Ralf, for giving me the opportunity to make progress in my personal and technical skills. You are the person who has supported me all the time – during my studies at the TUHH, at the time writing my master thesis, while offering me a position at your department, and during the last years investing time and effort to improve my skills in writing academic papers. I will never forget our late discussions at the university and on the phone.

To Nils, thank you for maintaining the technical infrastructure I have used for evaluating approaches. To Angela, thank you for supporting me in all organizational topics. Özgür, thank you for supporting me during a difficult time and motivating me in writing papers — I really enjoyed our coffee breaks and conversations at the university. Tanya, thank you for spending your time in writing papers talking about problems and new approaches, having coffee breaks and talking about everyday topics. Karsten, and Mona, thank you for giving me an understanding of the academia. Thank you to Magnus, spending your limited time in discussions about new approaches and supporting me during examination periods. Thank you to Marisa for our inspiring discussions during the journeys to the university and to Hamburg. I would also like to thank Marcel, Maurice, Mattis, Nils, and Simon for the great time at IFIS.

The past few years I have spend most of my time on my academic education as well as on the growth of my company, which was only possible thanks to my lovely family. Thank you to my parents for having always been there when I needed you and supporting me in all my project. You are the best parents I could have ever asked for. Thank you to my wonderful wife and mother of my children, accepting the focus of the last years and the limited time we had have during the past years. Thank you for supporting me in my projects and managing our everyday life. I am looking forward to a new chapter

in my life spending time with my family.

This work closes a special chapter of my life and I will never forget the exhausting and great time I have had the last past years.

Thank you!

Felix Kuhr Lübeck, October 2021

Contents

List of Algorithms

List of Abbreviations

ERV Expected Relevance Value
FDR False Discovery Rate
FN False Negative
FP False Positive
HMM Hidden Markov Model
IE Information Extraction
iSCD inline Subjective Content Description
IR Information Retrieval
KB Knowledge Base
LDA Latent Dirichlet Allocation
LSI Latent Semantic Indexing
ML Machine Learning
MPSCD Most Probably Suited Subjective Content Description
MUC Message Understanding Conference
NE Named-Entity
NEL Named-Entity Linking
NELL Never Ending Language Learner
NER Named-Entity Recognition
NLP Natural Language Processing
PLSI Probabilistic Latent Semantic Indexing
PoI Positions of Interest
POS Part-of-Speech
PPV Positive Predictive Value
RDF Resource Description Framework
RTM Relational Topic Model
SCD Subjective Content Description
SRL Statistical Relational Learning

TN True Negative

TP True Positive

TPR True Positive Rate

List of Symbols

w	Word
sen	Sentence
V	Vocabulary
d	Document
$\mathcal{D}$	Corpus
$\mathcal{M}(\mathcal{D})$	Topic model from documents in corpus $\mathcal{D}$
K	Number of topics in a topic model
$\mathcal{K}(\mathcal{M}, \mathcal{M}')$	Similarity between topic models $\mathcal{M}$ and $\mathcal{M}'$
$z_{d,j}$	Topic for j-th word in document d
α	Hyperparameter for document-topic probability distributions
β	Hyperparameter for topic-word probability distributions
$Dir(\alpha)$	Dirichlet distribution from α
$Dir(\beta)$	Dirichlet distribution from β
θ	Document-topic probability distribution in LDA
ϕ	Topic-word probability distribution in LDA
$\mathcal{D}_d$	Subcorpus optimized for document d
χ	Link probability between documents in RTM
ι	Sigmoid function in RTM
ν	Hyperparameter for link probability in RTM
η	Weight vector for logistic regression in RTM
$\circ$	Hadamard product
H	Hellinger distance
τ	Hellinger distance threshold
J	Jaccard coefficient
$win_{d,\rho}$	Window in d with center at position ρ
σ	Window size
l^-	Window decreasing left
r^-	Window decreasing right
l^+	Window increasing left
r^+	Window increasing right
t	Subjective content description (SCD)
$T(\mathcal{D})$	Set of SCDs associated with documents in $\mathcal{D}$
$T(d)$	Set of SCDs associated with document d

δ	SCD word probability distribution
$I(w^d, win_{d,\rho})$	Influence value of SCD for word w^d in $win_{d,\rho}$
ERV_t^d	Expected relevance value of SCD t for document d
$\mathcal{M}_T$	Similarity score matrix for SCDs
sim	Similar document category
$unrel$	Unrelated document category
ext	Extending document category
rev	Revision document category
$\mathcal{C}$	Set of document categories
$\mathcal{W}$	Sequence of triples each containing SCD, SCD similarity value, and window details
λ	Hidden Markov Model
$\mathcal{H}$	Set of Hidden Markov Models
s	Hidden state
Ω	Set of hidden states
S	Hidden state sequence
y	Observation
O	Observation sequence
Δ	Set of observable states
A	Transition probability matrix
B	Emission probability matrix
π	Initial state distribution vector

Chapter 1

Introduction

Natural Language Processing (NLP) combines fields from linguistics, computer science, and artificial intelligence, and is an important discipline concerned with the interaction between machines and human language to process natural language data such as text. One famous branch in the domain of NLP is Information Extraction (IE), which is the task about structuring unstructured data and opens up new avenues to automatically extracting data from unstructured machine readable data such as text in documents. In the late 1970s, DeJong (1979) have introduced FRUMP — one of the first IE systems focusing on the textual context of news stories instead of only words in the news stories.

Generally, extracting information from text for a specific task is difficult since some kind of text understanding is necessary to interpret the text of a document. Thus, the U.S. government-sponsored Message Understanding Conference (MUC) encourages the development of new and better methods of IE. Over the last three decades, researchers have introduced different tasks trying to extract data from the text of documents, e.g., Named-Entity Recognition (NER), introduced by Grishman and Sundheim (1996), which is the discipline of extracting and tagging entities in a text, or relationship extraction, introduced by Chinchor (1998), which is the task of detecting and classifying relationships between entities. Other important tasks are sentiment analysis or detecting synonyms, irony, and sarcasm.

A variety of document-oriented information extraction approaches focus on techniques extracting data from the text of documents to (i) store the data in a structured format such that those data can be easily processed for different tasks, (ii) mine the emotions within the text, known as *sentiment analysis*, or (iii) add *interpretative linguistic data* to documents, e.g., by indicating the word class whereto the words in a text belong, such that words can be distinguished having the same spelling but different meanings. These words are also known as homonyms.

Generally, interpretative linguistic data are the product of text understanding and a specific task. Different types of linguistic data are defined such as Part-of-Speech (POS) tagging or *semantic annotations*, where POS tagging adds grammatical tags like noun, verb, or article to words. Those grammatical tags are necessary to understand the content of text. Semantic annotations add data about the semantic category of words which is useful to distinguish words having no difference in spelling or pronunciation, but a

different semantic meaning. Generally, adding data to documents should increase the performance of a specific task, e.g., the performance of a document retrieval system such that more relevant documents might be identified for a given query. Adding interpretative linguistic data to an individual collection of documents is known as *corpus annotation* and is widely accepted as one of the key contributions to the benefit of a corpus, since interpretative linguistic data add a value for different tasks performed on the individual collection of documents representing the corpus. Thus, the performance of retrieval systems may increase for an annotated corpus containing not only data explicitly available within the text of documents, but containing additional data about the documents' content. Mostly every modern search engine is based on the document retrieval task where queries of users are matched against available documents. We use the term *add a value* to describe the performance increase for a specific task on annotated documents compared to unannotated documents.

In this dissertation, we follow the agent-theoretic perspective of Russell and Norvig (2009), where an agent is a rational, autonomous unit acting in a world, perceived through sensors, fulfilling a task, e.g., adding data to documents, or providing document retrieval services given specific requests from users. We use the term *agent* referring to a software agent for an information retrieval service and assume that an agent is working with an individual collection of documents acting as a reference library. We assume that an agent is working with an individual collection of documents pursues a defined task or task, e.g., providing an information retrieval system matching a given query against a set of documents and returns a set of best matching documents. The agent might not only use the words available in documents and queries trying to match documents to a query but annotate documents with Subjective Content Descriptions (SCDs), which the agent expects to be relevant to its corpus-specific task. Maybe the most intuitive way to imagine an SCD is given by a human generated *Post-it Note* associated with a document since the content of a Post-it Note is subjective and task-specific. Generally, we assume that the task provides a context in which SCDs add value for the task of an agent, e.g., notes added to specific sentences of an article may provide explanations or references. Thus, SCDs add data relevant for the agent's task and have a connection to a specific section in the document s.t SCDs may optimize the performance of an agent's retrieval service.

The problem this dissertation focuses on is that of *estimating context-aware subjective content descriptions for documents adding a value for corpus-specific tasks of an agent,* where a corpus-specific task might be identifying documents in a given corpus that are similar to a new document presented to an agent's retrieval system or identifying the corpus-driven category of a new document an agent is faced with.

Available annotation systems enriching documents in a corpus with annotations focus only on the content of documents and ignore the task of an agent as well as the individual collection of documents within a corpus representing a specific context the agent is acting

in. In contrast to available annotation systems, the goal of this dissertation is to design techniques resulting in context-aware and subjective content descriptions for documents by considering the individual collection of documents in a corpus given a corpus-specific task. Next, we give an overview of related work in the domain of corpus annotation and IE. Afterwards, we present the research objectives and scientific contributions of this dissertation, followed by a short overview of all chapters.

1.1 Related Work

In this section, we present related work in the domain of corpus annotation, which is a sub-task in the field of corpus linguistics. Corpus annotation can be defined as the process of adding *interpretative* and *linguistic data* to documents in a corpus. Generally, the granularity of annotations depends on the application and a single annotation may cover a word, a sentence, a paragraph, a document, or an entire corpus. We refer to ISO-Standard:24612:2012 (2012) for further details about the granularity of annotations. Electronic corpus linguistics dates from the work of Henry Kučera and W. Nelson Francis at Brown University in 1978. They have presented the first electronic corpus that has been generated from publications consisting just over one million words from 500 text samples of about 2000 words each.

Over the recent years, researches from the NLP community have introduced a considerable number of annotation systems to enrich documents in a corpus with additional data, denoted as annotations. The annotation systems can be grouped into the following three categories:

(i) manual annotation systems,

(ii) semi-automatic annotation systems, and

(iii) automatic annotation systems.

An important research goal is automatically enriching documents in a corpus with context-specific data adding a value to the task of an agent, e.g., for retrieval tasks of an information retrieval agent working with an individual collection of documents in a corpus. Generally, the value of data that has been added to a document depends on the specific task of an agent. However, to control the *quality* of the annotations enriching a document, most of the non-manual annotation systems are only semi-automatic instead of fully automatic.

Manual Annotation Systems Manual annotation systems require human annotators adding data to documents. One advantage of manual annotation systems is that new data associated with a document might have a high quality since often domain-specific annotation experts, having background knowledge of the documents' content, would

manually annotate documents in a corpus. However, manually enriching documents with annotations is a time-consuming task, because each document has to be read by at least one human annotator generating annotations for the documents in a corpus. The quality of annotations depends on the background knowledge of the human annotator. Even if two domain experts annotate the same document, it is very unlikely that the documents share the same annotations since enriching documents with annotations is highly subjective and no single *correct* annotation exists. Thus, one of the greatest challenges in the domain of manual annotation systems is creating high-quality annotations such that the annotations adding a value for each specific task, e.g., increasing the information retrieval performance of an agent.

Generally, domain-specific annotation experts are scarce and quite expensive. Thus, another annotation approach to obtain manually generated annotations for documents is crowdsourcing (Biemann (2013); Bontcheva *et al.* (2014); Sabou *et al.* (2014)). Crowdsourcing focus on fast generation of annotations at relatively low cost. Jurgens (2013) considers that annotating documents in a crowdsourced fashion often suffers from high error rates, but using voting systems the error rates can be minimized, thereby yielding in an acceptable quality of annotations. Annotations generated in a crowdsourced fashion are more generic, since human annotators are no domain-specific experts and have only limited domain knowledge. Generally, depending on the specific task, annotations generated in a crowdsourced fashion might be sufficient.

Next, we present some manual annotation systems for text documents.

Cunningham (2002) has introduced GATE, a general architecture for text engineering. GATE Teamware (Bontcheva *et al.* (2013)) is part of the GATE family and provides a collaborative annotation environment for manually annotating text by highlighting words or completing sentences. Although there is an advantage in the use of GATE Teamware, that other parts of the software family can be easily used, as well.

Stenetorp *et al.* (2012) have introduced Brat, a famous web-based tool for structured annotation of text. In Brat, the annotations are not freeform text but have a fixed form, such that the annotations can be automatically processed and interpreted by a machine and different human annotators can easily create annotations of the same form. Brat aims to support manual curation efforts and increase annotator productivity by providing powerful annotation functions and rich visualization ability.

Eckart de Castilho *et al.* (2016) have introduced WebAnno, another web-based tool for text annotation, to be prominent by their annotation project management, freely configurable set of tags, and user management. WebAnno supports different types of documents using an import function and provides part-of-speech tagging, named entity extraction, dependency parsing, and co-reference chains. Additionally, WebAnno provides an interface to farm out the task of annotations to a crowdsourcing platform.

Yang *et al.* (2018) have introduced YEDDA, a lightweight, efficient, and comprehensive open-source tool for text annotation including collaboration, evaluation of annotations, and annotation analysis. In comparison to most annotation systems, YEDDA provides

batch annotation for entities using a command line interface and presents recommendations for text annotations. Additionally, the system supports shortcut keys to efficiently annotate text by hand. Those features reduce the required time for human annotators adding annotations to text documents.

Automatic Annotation Systems In comparison to manual annotation systems, automatic annotation systems analyse the text of a document whilst trying to identify the context and automatically enriching a document with additional data. A well-established automatic annotation task is POS tagging, which is the task of taking an unannotated block of text and producing an annotated block of text that highlights the names of entities and classifies the entities into predefined categories such as persons, organizations, or locations. Some automatic annotation systems take the identified Named-Entitys (NEs) from the text and link those entities to an external source by relating more linked entities and relationships which can be identified using link prediction. Getoor and Diehl (2005) describe link prediction as the discipline of estimating the likelihood of the existence of a link between nodes, using the given links and attributes of nodes within a graph.

The quality of data from external sources, acting as annotations for a given document, depends on the document's content and applicability of the source for the content of documents. Generally, annotations are generic and mostly ignore the context presented by the corpus. Two famous sources used to link entities to a document are DBpedia and YAGO introduced in Lehmann *et al.* (2014) and Suchanek *et al.* (2007), respectively.

Day *et al.* (2000) have introduced cross-document annotation, which is a framework that serves as a corpus-wide Knowledge Base (KB) for linguistic annotations. The framework extracts data about individuals and events from different textual sources within the same corpus to accrue to a single representation of that individual. The goal of the framework is to support human annotators to accurately, and rapidly find previously annotated candidate mentions and facilitate inspection of those mentions in context.

Dill *et al.* (2003) have introduced SemTag to generate semantic annotations. SemTag is a module of the large-scale text analytics platform Seeker. SemTag uses a structural analysis of text and the ontology TAP, introduced by Guha and McCool (2003), to automatically annotate documents with data from the TAP ontology, containing lexical and taxonomic information about movies, authors, locations, musicians, and others. SemTag stores the annotations in a separate file.

Analogous to SemTag, the semantic annotation platform KIM, introduced by Popov *et al.* (2003), provides a knowledge and information management infrastructure for automatic semantic annotation, indexing, and retrieval. KIM identifies entities in the text of documents and links the entities to semantic descriptions which are provided by the KIMO ontology, which is a pre-populated ontology with many instances. The KIM platform allows KIM-based applications for automatic semantic annotation. KIM is based on GATE and has been applied in different domains like anti-corruption, asset recovery,

analysis of biomedical content, or scientific papers.

Reuters has introduced OpenCalais[1], which has been a knowledge extraction tool and has been offered as a web service by Reuters, which automatically tags data in unstructured text using a large ontology. OpenCalais extracts NEs with sense tags, such as people, companies, books, albums, facts, and events. The annotations cover words. Different plugins are available like the OpenCalais TopBraid Composer, which generates automatic RDF files. Reuters has sold OpenCalais some years ago to Refinitive.

Yosef *et al.* (2011) have introduced AIDA, which is an online tool for accurate disambiguation of NEs in text and tables. AIDA generates for a given text a mapping from ambiguous names to canonical entities like people or places that are registered in an available external ontology. AIDA has a web-based online interface accepting different input formats and returning proper entities, e.g., for the input *UC Berkeley*, AIDA returns *University of California, Berkeley* as output.

Petasis *et al.* (2011) have introduced BOEMIE, which is another ontology-based text annotation tool, focusing on text block locations that correspond to specific types of NEs, and additionally performs annotations of text that refers to the same domain. BOEMIE automatically creates annotations from HTML files or text documents and stores them in a separate file. BOEMIE supports manual annotations, too.

For further details about existing annotation systems please refer to the surveys of Oliveira and Rocha (2013), Gangemi (2013), and Liao and Zhao (2019).

Information Extraction Automatic annotation systems often rely on IE techniques extracting structured data from unstructured text, e.g., in the form of relational tuples. The tuples consist of two arguments and a phrase denoting the relation between both arguments. Traditional approaches extracting data from the text of a document have focused on the well-defined query answering tasks over a predefined set of target relations on a small set of homogeneous documents by taking a target relation with hand-crafted extraction patterns. Those hand-crafted rules work fine for a specific domain and traditional IE requires extensive human involvement. Yates *et al.* (2007) have introduced *TextRunner*, which is one of the first *IE* systems relying on unsupervised extraction strategies reducing the manual effort for extraction patterns or training data to a minimum. Those kind of *IE* systems are known as *OpenIE* systems. Using unsupervised extraction strategies requires only few extraction patterns which need to be manually defined or a small set of training data.

Having extracted entities from the text of a document allows an agent to identify those entities in external sources. Afterwards, the agent might enrich documents with additional data having a relationship with those extractable entities. As mentioned before, annotations enrich a document with additional data, e.g., to distinguish words having no difference in spelling or pronunciation, but a different semantic meaning. Thus,

[1] `https://www.refinitiv.com/en/products/intelligent-tagging-text-analytics`

OpenIE is an important discipline to automatically enrich documents with data.

In recent years, systems have emerged using methods in the domain of IE and Statistical Relational Learning (SRL) to extract data from text of millions of randomly selected unstructured documents and derive large graphs, representing a symbolic content description using entities and relations between entities. Some of the most known systems containing relational data are DeepDive (Zhang (2015)), Never Ending Language Learner (NELL)(Carlson *et al.* (2010)), YAGO (Suchanek *et al.* (2007)), FRED (Presutti *et al.* (2012)), and KnowledgeVault (Dong *et al.* (2014)).

Generally, manual and automatic annotation systems are helpful to enrich a document with additional data, supporting an agent or even humans on a corpus-specific task for documents. However, we take the view that separately annotating documents ignores the individual collection of documents represented by a corpus. Additionally, available annotation systems ignore the available annotations associated with other documents within the annotation process of an unannotated document and enrich documents with general data from external sources by matching extractable entities from the text of the documents to entities in the external sources to associate externally available entity-related data to the documents.

Enriching documents with externally available entity-related data might increase the performance of Information Retrieval (IR) tasks, since documents contain more data related to entities covered by the documents' text. However, those data ignores the specific context represented by the individual collection of documents in a corpus and the text of the document itself and only focus on the NEs extractable from the text of documents.

Thus, this dissertation lays the foundation for subjective content descriptions for documents adding a value for corpus-specific tasks of an agent.

1.2 Research Objectives and Scientific Contributions

Generally, in the last decade, many efforts have been made in extracting structured data from text in publicly available documents to estimate semantic relationships between entities. Compared to the research of the last decade, the goal of this dissertation is to lay the foundation of automatically annotating documents in a corpus with subjective content descriptions to support an information retrieval agent on its corpus-specific tasks, e.g., identifying a set of documents in a corpus being similar to a new document or estimating SCDs for a new document based on the SCDs associated with documents available in a corpus.

In detail, the objective of this dissertation is to provide techniques such that:

(1) Annotated documents in a corpus can be used to enrich an unseen corpus-extending document with corpus-driven descriptions in form of SCDs.

(2) An agent might be interested in extending its corpus only with documents having certain properties represented by different document categories. A category might represent the similarity of a new document to documents in the corpus. Based on selected SCDs associated with a new document an agent can identify the document category with respect to the documents in a given corpus and use the associated SCDs directly for additional tasks.

(3) SCDs associated with documents in one corpus can be used to enrich documents from another, but related corpus.

We introduce annotation techniques based on SCDs to bridge the gap between advantages of manual annotation systems providing high quality annotations for documents at high cost and advantages of automatic annotation systems generating annotations for documents without considering the individual collection of documents in a corpus.

In this dissertation, we make several contributions to subjective content descriptions and we can summarize the contributions as follows:

(1) Context-specific Corpus Enrichment An agent working with a reference library containing an individual collection of documents is a common setting in an information retrieval scenario. Over time, the agent is faced with new documents and an important question for an agent in the domain of information retrieval is: *Does the content of a new document provides anything of value to add in the context of the reference library?* We show the potential of augmenting and automatically enriching a reference library with new documents while considering the context represented by the reference library.

(2) Corpus-driven SCD Enrichment Manually generating SCDs and associating them to documents in a corpus is a time-consuming and expensive task, since domain-specific annotation experts have to read and annotate all documents in a corpus. Given a corpus containing at least sparse and weakly annotated documents, we present an approach to enrich those documents with additional SCDs associated with related documents in the same corpus. Thus, an agent might benefit from those additional SCDs.

This contribution comprises **(a)** a definition of two similarity values for documents resulting in **(b)** a definition of a relevance value for SCDs, and **(c)** an unsupervised iterative annotation approach. The approach uses a combination of both similarity values, as well as the relevance values of SCDs to enrich sparse and weakly annotated documents with SCDs of related documents within the same corpus.

(3) Identifying SCDs within Text An agent in pursuit of new documents may come across with documents where *normal* document text, i.e., content, and textual content descriptions, i.e., SCDs, are interleaved. Manually identifying the SCDs within text or creating a parser for separating the SCDs from the content by manually specifying rules

for distinguishing content and SCDs is cost-intensive and requires intimate knowledge about content and SCDs. We present an approach to automatically identifying SCDs within text by using the SCD-word probability distribution of documents in a corpus, which encodes as hidden state for each word in the document whether the word is part of an SCD or not. Calculating a most probable sequence of hidden states in a Hidden Markov Model (HMM) then allows for identifying the most probable sequence for SCDs in text.

(4) Context-aware Adaptation of SCDs SCDs associated with documents in one corpus, might add a value for the agent's task working with documents in another corpus. Automatically associating SCDs associated with documents in one corpus to documents in another corpus while considering the context-shift between both corpora is an important task, since manually generating SCDs for documents is expensive.

We present an unsupervised domain adaptation approach adapting the SCD-word probability distribution from a source corpus to a target corpus supporting an agent with an initial set of context-aware SCDs for documents in a non-annotated target corpus, such that an agent has an initial set of SCDs.

(5) Maintaining Topic Models for Growing Corpora The individual collection of documents in a corpus might change over time since an agent decides to extend its library with additional documents. Changing the documents in a corpus results in a different similarity between documents and a modified relevance value of each associated SCD. We analyse different techniques for different scenarios to incorporate corpus-extending documents into a given topic model resulting from the initial documents in a corpus, such that an agent can identify for the corpus-extending documents a set of similar documents in the entire corpus based on the documents' topic similarity.

(6) Named-Entity Induced Links for Relational Topic Models Relations between documents provide important data for topic models. The text of similar documents might share NEs and we can assume that documents sharing an entity are connected to each other. We use the Relational Topic Model (RTM) approach, introduced by Chang and Blei (2009), representing the relationship between documents in a corpus by adding a link between two documents sharing an entity.

We **(a)** estimate the properties of NEs improving the performance of the RTM and **(b)** compare the performance between the well-known topic modeling approaches LDA and RTM by considering different settings for entity induced links between documents in a corpus.

1.3 Structure

After this introduction, we start a chapter on preliminaries, presenting definitions, giving a brief overview of statistical models to identify hidden semantic structures in documents, and providing an overview of corpus annotation and information extraction. After the preliminaries, the main body of this dissertation begins, which is divided into two parts, presenting the contributions of this dissertation.

- Part I introduces SCDs and different techniques to add a value to a corpus by enriching documents with SCDs.
  - Chapter 3 presents SCDs itself along with most probably suited SCDs for documents. Additionally, the chapter introduces notations on corpus annotation (Contributions **1**).
  - Chapter 4 presents an approach for context-specific enrichment of a corpus with new documents by considering the category of the new documents (Contribution **1**).
  - Chapter 5 presents techniques to enrich documents in a given corpus with SCDs associated with related documents of the same corpus (Contribution **2**).
  - Chapter 6 presents an HMM-based approach to identify SCDs within texts and classify new documents into a predefined category (Contribution **3**).
  - Chapter 7 presents domain-adaptation techniques for SCD-word probability distribution to automatically associate SCDs associated with documents in one corpus to documents in another corpus with SCDs (Contribution **4**).

  This first part was mainly published in:

  Tanya Braun, Felix Kuhr, and Ralf Möller. Unsupervised text annotations. In *Formal and Cognitive Reasoning - Workshop at the 40th Annual German Conference on AI (KI-2017)*, 2017

  Felix Kuhr, Bjarne Witten, and Ralf Möller. Corpus-driven annotation enrichment. In *2019 IEEE 13th International Conference on Semantic Computing (ICSC)*, pages 138–141. IEEE, 2019

  Felix Kuhr, Tanya Braun, Magnus Bender, and Ralf Möller. To Extend or not to Extend? Context-specific Corpus Enrichment. In *Proceedings of AI 2019: Advances in Artificial Intelligence*, pages 357–368. Springer, 2019

  Felix Kuhr, Tanya Braun, Magnus Bender, and Ralf Möller. Augmenting and automating corpus enrichment. *Int. J. Semantic Comput.*, 14(2):173–197, 2020

Felix Kuhr, Tanya Braun, and Ralf Möller. Augmenting and automating corpus enrichment. In *2020 IEEE 14th International Conference on Semantic Computing (ICSC)*, pages 69–76. IEEE, 2020

Felix Kuhr, Magnus Bender, Tanya Braun, and Ralf Möller. Context-specific adaptation of subjective content descriptions. In *Proceedings of AI 2020: Advances in Artificial Intelligence*. Springer, 2020

Magnus Bender, Tanya Braun, Marcel Gehrke, Felix Kuhr, Ralf Möller, and Simon Schiff. Identifying subjective content descriptions among text. In *IEEE 15th International Conference on Semantic Computing, ICSC 2021, Virtual, January 27-29, 2021*, pages 451–458, 2021

Magnus Bender, Tanya Braun, Marcel Gehrke, Felix Kuhr, Ralf Möller, and Simon Schiff. Identifying subjective content descriptions among text. *will be published in: Int. J. Semantic Comput.*, 15(4), 2021

Felix Kuhr, Magnus Bender, Tanya Braun, and Ralf Möller. Context-specific adaptation of subjective content descriptions. In *IEEE 15th International Conference on Semantic Computing, ICSC 2021, Virtual, January 27-29, 2021*, pages 451–458, 2021

- Part II contains two additional chapters. Chapter 9 presents techniques for handling topic models representing a growing corpus and new documents (Contribution **5**). Chapter 10 presents results for the relational topic model using NE induced links as relation between documents (Contribution **6**).

The second part is based on the following publications:

Felix Kuhr and Ralf Möller. Constructing and maintaining corpus-driven annotations. In *2019 IEEE 13th International Conference on Semantic Computing (ICSC)*, pages 462–467, Jan 2019

Felix Kuhr, Magnus Bender, Tanya Braun, and Ralf Möller. Maintaining topic models for growing corpora. In *IEEE 14th International Conference on Semantic Computing, ICSC 2020, San Diego, CA, USA, February 3-5, 2020*, pages 451–458, 2020

Felix Kuhr, Matthis Lichtenberger, Tanya Braun, and Ralf Möller. Enhancing relational topic models with named entity induced links. In *IEEE 15th International Conference on Semantic Computing, ICSC 2021, Virtual, January 27-29, 2021*, pages 451–458, 2021

The first part and all subsequent chapters each end with a brief interim conclusion. Chapter 11 presents overall conclusions and takes a look into what the future might hold for subjective content descriptions.

Chapter 2

Preliminaries

This section starts with an introduction to notations before presenting an overview of information retrieval techniques and general definitions in corpus annotation.

2.1 Notations

We formalize the setting of a corpus representing an individual collection of documents, where documents might be associated with a set of additional data, i.e., SCDs.

Definition 2.1.1 (word)**.** A word w is a basic unit of discrete data from a vocabulary $\mathcal{V}=(w_1,\ldots,w_V)$, $V \in \mathbb{N}$, and can be represented as a one-hot vector of length V having a value of 1 where $w = w_i$ and 0's otherwise.

Example 2.1.1 (word)**.** Let us assume that we have the following eleven words representing w_1 to w_{11}, respectively: *David*, *Blei*, *received*, *the*, *ACM*, *Infosys*, *Foundation*, *Award*, *in*, *summer*, and *2013*. We can represent each of the eleven words by a one-hot vector, e.g., w_4 can be represented as $\langle 00010000000 \rangle$ and w_5 as $\langle 00001000000 \rangle$.

Definition 2.1.2 (document)**.** A document d is a sequence of words $(w_1^d,\ldots,w_N^d)$, $N \in \mathbb{N}$. Function $\#words(d)$ returns the total number of words in d, i.e., $\#words(d) = N$.

Example 2.1.2 (document)**.** A document is anything containing a sequence of words, e.g., books, journals, papers, or manuscripts. We assume that documents are electronic documents and that the documents are machine readable. Generally, techniques are available to create an electronic version from the non-electronic document. The following sequence of words represents a document:

> "David Blei received the ACM Infosys Foundation Award in summer 2013. He is known for latent Dirichlet allocation."

Definition 2.1.3 (sentence)**.** A sentence $sen \in d$ contains $Q \in \mathbb{N}$ words and is defined as a sequence of words: $sen = (w_1^d, w_2^d, ..., w_Q^d)$, where w_i^d represents the i-*th* word in *sen* of document d.

Example 2.1.3 (sentence)**.** The sentence *"David Blei received the ACM Infosys Foundation Award in summer 2013."* in document d contains eleven different words, namely *David*, *Blei*, *received*, *the*, *ACM*, *Infosys*, *Foundation*, *Award*, *in*, *summer*, and *2013*. We represent the sentence by a sequence words w_1^d to w_{11}^d:
$(w_1^d, w_2^d, w_3^d, w_4^d, w_5^d, w_6^d, w_7^d, w_8^d, w_9^d, w_{10}^d, w_{11}^d)$, where each word refer to an entry in a vocabulary $\mathcal{V}$.

Generally, the meaning of a word depends on the context the word occurs in. A well known form representing a specific context is a document containing a sequence of words. Another item representing a specific context is given by a corpus.

Definition 2.1.4 (corpus)**.** A corpus $\mathcal{D}$ represents a set of $D \in \mathbb{N}$ documents $\{d_1, \ldots, d_D\}$ and $\mathcal{V}_\mathcal{D}$ returns the corpus-specific vocabulary containing all different words occurring in the D documents of corpus $\mathcal{D}$.

Example 2.1.4 (corpus)**.** Collecting documents is not an end in itself and an agent collecting documents has a specific task in mind when collecting documents, e.g., collecting related work for a journal article or collecting books from the same author. Thus, the set of documents in a corpus are not randomly chosen and represents an individual collection of documents.

A corpus is not fixed and might change over time, e.g., an agent interested in extending the initial corpus with new documents relevant for the agent's task. We introduce document categories to classify new documents with respect to documents in a given corpus, s.t. the category of the same new document depends on the individual collection in a corpus.

Definition 2.1.5 (document category)**.** A new and previously unseen document $d' \notin \mathcal{D}$ can be classified based on the documents in corpus $\mathcal{D}$ using a classification function $classify_\mathcal{D}(d')$. We focus on the following four categories to classify a given document d' based on corpus $\mathcal{D}$: The content of a new document d' is

(i) similar to at least one document in $\mathcal{D}$ – category sim,

(ii) an extension of a document in $\mathcal{D}$ – category ext,

(iii) a revision of a document in $\mathcal{D}$ – category rev, and

(iv) unrelated to all documents in the corpus – category $unrel$.

The set containing all four document categories is defined as $\mathcal{C}$.

Example 2.1.5 (document category)**.** An agent might be interested in any of the four document categories depending on the specific task at hand. Let us assume, an agent provides a document retrieval service for a company and the corpus of the agent contains

only documents in a specific domain, e.g., the content of all documents is about NLP. The agent might be interested in extending the corpus with more documents containing text about NLPs. Thus, the agent is interested in documents from category *sim*.

Definition 2.1.6 (SCD)**.** An SCD t is a subjective content description and contains data. We can represent in a triple structure, s.t. $t = (s, p, o)$, where s represents a subject, p a predicate, and o an object. Generally, an SCD can take any form and in later chapters, we use a sequence of words as SCDs. The triple structure is similar to Resource Description Framework (RDF) and for specific tasks it is possible to represent SCDs as RDF triples. As such, the format of SCDs may be highly diverse and for our main contributions, the specific format of a subjective content description depends on the specific task of an agent working with the SCDs.

Example 2.1.6 (SCD)**.** An SCD t having the form of a triple is represented by a subject, a predicate, and an object. There exists no single perfect SCD for a sentence, since SCDs contain subjective descriptions about the content of documents, such that different SCDs are possible for the same document. The reason for different SCDs for the same sentence is given by the subjective interpretation of the corresponding content as well as the individual collection of documents in a corpus and the corpus-specific task of an agent.

An agent might build the following three SCDs (i) t_1, (ii) t_2, and (iii) t_3 for the following sentence in document $d \in \mathcal{D}$ *"David Blei received the ACM Infosys Foundation Award in summer 2013."*:

(i) t_1 : (David Blei, is, professor)

(ii) t_2 : (David Blei, teach at, Columbia University)

(iii) t_3 : (ACM Infosys Foundation Award, renamed into, ACM Prize in Computing)

However, another agent might build only a subset of those SCDs or another SCD, e.g., t_4 : (ACM Infoss Foundation Award, take place in, summer) depending on the task of the agent.

Definition 2.1.7 (located SCD)**.** An SCD t can be associated with a position ρ in a document $d \in \mathcal{D}$. We represent a located SCD t by the tuple $(t, \{\rho_i\}_{i=1}^{l})$, where $\{\rho_i\}_{i=1}^{l}$ represents the $l \in \mathbb{N}$ positions in document d that t is associated with.

Example 2.1.7 (located SCD)**.** In Example 2.1.6, we have mentioned that an agent might build SCDs (i) t_1 : (David Blei, is, professor) (ii) t_2 : (David Blei, teach at, Columbia University), and (iii) t_3 : (ACM Infosys Foundation Award, renamed into, ACM Prize in Computing) for document $d \in \mathcal{D}$ containing the sentence *"David Blei received the ACM Infosys Foundation Award in summer 2013."* Located SCDs extend SCDs with a location, such that each of the three SCDs t_1, t_2, and t_3 are associated with a position in the sentence.

Definition 2.1.8 (SCD set)**.** For each document $d \in \mathcal{D}$, there exists a corresponding *SCD set* $T(d)$. If document d is associated with located SCDs, then $T(d)$ contains a set of m^d located SCDs $\{(t_j, \{\rho_i\}_{i=1}^{l_j})\}_{j=1}^{m^d}$, and each SCD is associated with l positions in d. The set of all SCDs associated with documents in a corpus $\mathcal{D}$, ignoring the document-specific locations, is given by $T(\mathcal{D}) = \{t_j\}_{j=1}^{m}$, where $m = \sum_{d \in \mathcal{D}} m^d$.

Definition 2.1.9 (SCD window)**.** For each located SCD $t_j \in T(d)$ exists a corresponding SCD window $win_{d,\rho}$ referring to a sequence of words in d surrounding position ρ, i.e., $win_{d,\rho} = (w^d_{(\rho-i)}, ..., w^d_{\rho}, ..., w^d_{(\rho+i)})$, $i \in \mathbb{N}$ and ρ marks the middle of the window. The window-specific position of a word $w^d \in win_{d,\rho}$ is given by $pos(w^d, win_{d,\rho})$ and the size of window $win_{d,\rho}$ is given by $s(win_{d,\rho}) = 2i + 1$.

Example 2.1.8 (SCD window)**.** Let us assume SCD t_3 is associated with the sixth word position in Example 2.1.2. Then, we can create an SCD window $win_{d,\rho}$ for SCD t_3, which is associated with position $\rho = 6$. We underline the sixth word w_6 in d to visualize the associated SCD t_3.

$$win_{d,\rho} = (w_1, w_2, w_3, w_4, w_5, \underline{w_6}, w_7, w_8, w_9, w_{10}, w_{11}) \tag{2.1}$$

We argue that the distance between words in d and position ρ of an SCD $t \in T(d)$ represents the strength of a relationship between the SCD and the words in the window. Thus, we introduce an influence value to represent the relationship between words and SCDs more formally.

Definition 2.1.10 (influence value)**.** Each word w^d in the sequence of $win_{d,\rho}$ is associated with an influence value $I(w^d, win_{d,\rho})$ representing the distance between word position of w^d and the position ρ of the corresponding SCD. The closer a word is positioned to the position ρ in $win_{d,\rho}$, the higher its corresponding influence value is. Or in other words, the larger the distance between a word and the location of an SCD the smaller the influence value for the word. Generally, the function to estimate the influence value of a word depends on the specific task of an agent.

Example 2.1.9 (influence value)**.** Again, we use the same document as in Example 2.1.8 and estimate the influence value of a word in the window $win_{d,\rho}$. The influence value is similar to the binomial distribution for all words in the SCD window. The influence value of a word w^d at position $pos(w^d, win_{d,\rho})$ in document d is then given by:

$$I(w^d, win_{d,\rho}) = \binom{n}{k} \cdot q^k \cdot (1-q)^{n-k}, \tag{2.2}$$

where $n = s(win_{d,\rho})$, $k = pos(w^d, win_{d,\rho})$, and $q = \frac{\rho}{n}$. Thus, we can estimate the following influence values for $win_{d,\rho}$ in Expression (2.1).

$$(0.000965, 0.00977, 0.04395, 0.11719, 0.20508, \underline{0.24609}, 0.20508, 0.11719, 0.04395, 0.00977, 0.000965) \tag{2.3}$$

Evaluation Metrics

In this dissertation, we evaluate the performance of different techniques by using standard evaluation metrics. All metrics are based on True Positives (TPs), False Positives (FPs), True Negatives (TNs), and False Negatives (FNs).

A TP is an outcome where a model correctly predicts the positive class. Similarly, a TN is an outcome where a model correctly predicts the negative class. A FP is an outcome where a model incorrectly predicts the positive class. And a FN is an outcome where the model incorrectly predicts the negative class.

Definition 2.1.11 (Precision)**.** Precision, also denoted as Positive Predictive Value (PPV), attempts to answer the following question: What proportion of positive identifications was actually correct? In other words, precision represents the probability that a positive result is a true positive. Mathematically, precision is defined as follows:

$$\text{Precision} = \frac{\#TP}{\#TP + \#FP} \tag{2.4}$$

Definition 2.1.12 (Recall)**.** Recall, also denoted as true positive rate (TPR), attempts to answer the following question: What proportion of actual positives was identified correctly? Mathematically, recall is defined as follows:

$$\text{Recall} = \frac{\#TP}{\#TP + \#FN} \tag{2.5}$$

Definition 2.1.13 (F_1-score)**.** The F_1-score is the harmonic mean of the precision and recall and is defined as follows:

$$F_1\text{-score} = \frac{2 \cdot \text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}, \tag{2.6}$$

Definition 2.1.14 (Accuracy)**.** Accuracy is a metric for evaluating classification models. Informally, accuracy is the fraction of numbers of predictions a model got right. Formally, accuracy has the following definition:

$$\text{Accuracy} = \frac{\#TP + \#TN}{\#TP + \#TN + \#FP + \#FN}, \tag{2.7}$$

where # refers to the number of occurrences of an event, i.e. TP, FP, TN, and FN.

Definition 2.1.15 (False Discovery Rate)**.** The False Discovery Rate (FDR) is the expected ratio of the number of false positive classifications to the total number of positive classifications. Mathematically, FDR is defined as follows:

$$FDR = \frac{\#FP}{\#FP + \#TP}. \tag{2.8}$$

In Chapter 9 and Chapter 10, we evaluate the performance of a topic model using the perplexity measure on held-out data given by Newman *et al.* (2009).

Definition 2.1.16. The perplexity is a measurement of how well a probability distribution predicts a sample and can be used to to compare different probability models with each other.

Mathematically, the perplexity is defined as follows:

$$perplexity(w) = \exp\left(-\frac{\log\left(p(w)\right)}{\sum_{d=1}^{D}\sum_{j=1}^{V} n^{(jd)}}\right), \tag{2.9}$$

where $n^{(jd)}$ represents the occurrence of the j-th word in document d. The smaller the perplexity, the better the quality of the model, since a low perplexity indicates the probability distribution is good at predicting the sample.

Next, we give an overview in the domain of IR systems and present techniques to compare documents with each other.

2.2 Information Retrieval

Information retrieval is the process of obtaining data from an IR system that are relevant to the information need of an agent and the agent might search for information in a document or for documents themselves. In the 1980s, many document retrieval methods were based on word matching strategies to identify best matching documents for natural language queries. Those retrieval methods represent documents as a bag of words by ignoring the specific sequence of words within documents. The bag of words representation simplifies the documents' structure.

Generally, the basic scenario for document retrieval methods was an agent that formulates a query by providing a small set of keywords and expects a document retrieval system to return those documents from a corpus, where the corpus contains data the agent is interested in. However, document retrieval methods based on word matching strategies are often too simple. The problem with word matching strategies is that agents want to retrieve information on the basis of the core sense of a word which is also known as the conceptual meaning. However, word matching strategies ignore the conceptual meaning and use only the words themselves and individual words cannot provide evidence about the conceptual meaning of a document since usually many ways exist to express a given concept such that query terms in an agent's query may not match with the words of relevant documents. Additionally, most words have multiple meanings s.t. an agent's query will match to terms in documents that are not of interest to the agent.

Latent Semantic Indexing. Deerwester *et al.* (1990) have introduced Latent Semantic Indexing (LSI), which is an approach for mapping documents and query terms into a latent semantic space by performing a dimension reducing linear projection based on singular value decomposition (SVD). First, the authors create for documents in a corpus a document-term matrix containing word counts per document, where rows represent words and columns represent documents. The SVD technique reduces the number of rows in the document-term matrix while preserving the similarity structure among columns such that words with similar conceptual meaning are represented in the same way. The latent semantic space allows more reliably estimating the similarity between documents, e.g., by using the cosine of the angle between two document vectors. The document vectors are given by the columns of the document-term matrix. For IR, the query of an agent is translated into the same low-dimensional space as performed for the documents in a corpus. Afterwards, similar documents can be identified for a query by using the cosine of the angle between the vector representation of both, documents and the query. However, LSI has no probabilistic model of term occurrences and the results are difficult to interpret.

Probabilistic Latent Semantic Indexing. Thus, some years later, Hofmann (1999) has introduced the Probabilistic Latent Semantic Indexing (PLSI) model which is a latent variable model based on the idea of a latent class model, where a set of observed multivariate variables relates to a set of latent variables. In the PLSI model, the observable words from documents mapped to c hidden classes. The classes are denoted as *topics*. PLSI models the probability of each co-occurrence of words and documents as a mixture of conditionally independent multinomial distributions. In terms of a generative model, which is a statistical model of the joint probability distribution, PLSI is defined in the following fashion:

(i) Select document d with $p(d)$,

(ii) take a latent class c with $p(c \mid d)$,

(iii) generate a word w with $p(w \mid z)$.

Each document in a corpus is characterized by a specific mixture of factors with weights $P(c \mid d)$. This means, that each document is represented by a probability distribution over the c hidden classes. Figure 2.1 presents a graphical representation of PLSI.

Today, Machine Learning (ML)-based text-mining approaches trying to identify hidden semantic structures within the text of documents relating a set of observed multivariate variables to a set of hidden classes are denoted as *topic models* — thus, PLSI is one of the first topic modeling approaches. The hidden classes resulting from a topic modeling approach emerge directly from the analysis of words in a collection of documents.

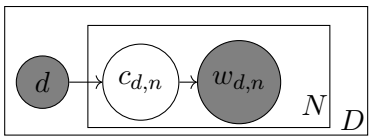


Figure 2.1: Graphical representation of PLSI. The boxes represent replicates for the (D) documents and the (N) words. Variable c is hidden; d and w are visible.

However, topics are only a synonym for hidden classes and no high-level descriptions like *sports*, *places*, or *film-industry* are available for topics.

Generally, the results of PLSI have a clear probabilistic interpretation and the problems like polysemi of words are better addressed than in word matching strategies and the LSI approach. However, the PLSI model is incomplete as it provides no probabilistic model at the document level. In PLSI, each document is represented as a list of numbers and no generative probabilistic model is given for these numbers such that the number of parameters in the model grows linearly with the size of the corpus, leading to overfitting. Additionally, it is unclear how to assign a probability to documents not part of the training set. Thus, some years later, Blei *et al.* (2003) have introduced LDA focusing on the deficiencies in the PLSI model.

2.2.1 Latent Dirichlet Allocation

Blei *et al.* (2003) have introduced the LDA topic modeling approach as a generalization of the PLSI model. The techniques used in LDA are similar to PLSI, except that LDA makes use of a sparse Dirichlet prior for the document topic probability distribution and another Dirichlet prior for the topic word probability distribution.

Similar to PLSI and LSI, Blei *et al.* (2003) follow the idea to represent the documents as a bag of words to simplify the structure of documents by ignoring the sequence of words within the text of documents. Blei *et al.* (2003) assume documents to be represented as a mixture of topics and each topic is characterized by a probability distribution of words from the corpus-driven vocabulary containing all different words occurring in the documents of a given corpus $\mathcal{D}$. The document topic probability distribution can be interpreted as a dimensionality reduced representation of a document and the dimensionality is given by the number of topics (K) used in the topic model. For a document d in a given corpus $\mathcal{D}$, the LDA approach learns a discrete probability distribution θ_d. The probability distribution θ_d contains for each topic $k \in \{1, \ldots, K\}$ a value between 0 and 1 and for each document $d \in \mathcal{D}$ the sum over all K topics is 1.

A corpus $\mathcal{D}$ contains documents $d \in \{d_1, ..., d_D\}$, where each document d is represented as a set of words $w_d \in \{1, ..., N_d\}$. A per-word topic assignment $z_{d,n}$ is drawn from the per-document topic proportion vector θ_d. Each topic $k \in \{1, ..., K\}$ is a multinomial probability distribution of the vocabulary generated from all documents in $\mathcal{D}$.

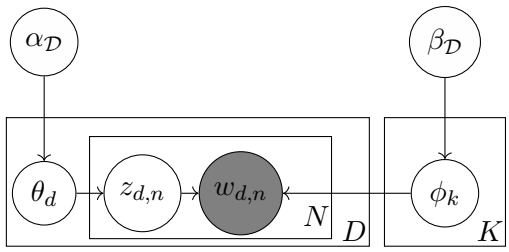


Figure 2.2: Graphical representation of an LDA topic model. The boxes represent replicates. Variables ϕ_k, θ_d, and $z_{d,n}$ are hidden variables; $\alpha_{\mathcal{D}}$ and $\beta_{\mathcal{D}}$ are corpus-specific hyperparameters and the N words $w_{d,n}$ (shaded) within the D documents are visible.

Additionally, the LDA model has two corpus-specific hyperparameters α and β, where $\alpha_{\mathcal{D}}$ conditions the per-document topic probability distribution θ_d and $\beta_{\mathcal{D}}$ conditions the per-corpus topic probability distributions ϕ_k, $k \in \{1, ..., K\}$ for corpus $\mathcal{D}$.

The two hyperparameters trade off the following two goals to find groups of tightly co-occurring words:

(i) Allocate the words of documents to as few topics as possible.

(ii) Assign a high probability to as few terms as possible in each topic.

However, these two goals are conflicting. Assigning all words within a document to a single topic achieves the first goal, but makes it difficult to achieve the second goal. Achieving the second goal and assigning only few words to each topic makes it difficult to reach the first goal for documents containing many words.

Figure 2.2 presents an LDA topic model as a graphical model in plate notation. The boxes represent replicates and observable variables are shaded.

The generative process of LDA is defined as follows:

(1) Choose $\theta_d \sim Dir(\alpha_{\mathcal{D}})$, where $d \in \mathcal{D}$ and $Dir(\alpha_{\mathcal{D}})$ is a Dirichlet distribution with hyperparameter $\alpha_{\mathcal{D}}$.

(2) Choose $\phi_k \sim Dir(\beta_{\mathcal{D}})$, where $k \in \{1, ..., K\}$ and $Dir(\beta_{\mathcal{D}})$ is a Dirichlet distribution with hyperparameter $\beta_{\mathcal{D}}$.

(3) For each of the word positions j within each document $d \in \mathcal{D}$ and $j \in \{1, ..., N\}$, N is the total number of words in document d:

  (a) Choose a topic $z_{d,j} \sim Multinomial(\theta_d)$

  (b) Choose a word $w_{d,j} \sim Multinomial(\phi_{z_{d,j}})$

The multinomial probability distribution in step (3a) and step (3b) refers to the *categorical probability distribution*, since the authors of LDA use a multinomial with only one trial by selecting a topic in step (3a) and a word from the selected topic in step (3b).

Generally, for documents in a corpus nothing but words are visible. At the beginning no per-topic word probability distribution, or per-document topic probability distribution is available. The key inference problem in the LDA topic modeling approach is computing the posterior distribution that refers to reversing the generative process and learning the posterior distributions of latent variables in the model given the observed data.

Mathematically, we can represent the inference problem computing the posterior distribution as following equation:

$$p(\theta, \phi, z \mid w, \alpha, \beta) = \frac{p(\theta, \phi, z, w \mid \alpha, \beta)}{p(w \mid \alpha, \beta)}, \tag{2.10}$$

where the left side of Expression (2.10) defines the probability of (i) the per-document topic probability distribution θ, (ii) the per-topic word probability distribution ϕ, and (iii) the topic labels z, given all words w and hyperparameters α and β.

Given K topics, hyperparameters α and β, we are interested in determining the per-document topic probability distribution θ_d for each $d \in \mathcal{D}$, the per-topic word probability distribution ϕ, and the word topic assignment z. However, exactly calculating the posterior distribution is intractable, since the normalization factor $p(w \mid \alpha, \beta)$ cannot be computed exactly since we have to marginalize over the hidden variables (Darling (2011)).

Different inference algorithms have been introduced which can be used to approximate the posterior distribution in Expression (2.10). Some approximative inference algorithms for the posterior distribution of hidden variables using expectation propagation (Minka and Lafferty (2002)) or Gibbs sampling (Griffiths and Steyvers (2004); Griffiths (2002); Pritchard *et al.* (2000)). Generally, Gibbs sampling is a special case of the Markov-chain Monte Carlo (MCMC) approach, where the dimensions of the distribution are sampled alternately one dimension at a time, s.t. the fixed dimension is conditioned on the values of all other dimensions. For further details about the Gibbs sampling algorithm we refer to Griffiths (2002) and Griffiths and Steyvers (2004). Blei *et al.* (2003) have introduced a variational inference algorithm making use of Jensen's inequality to obtain an adjustable lower bound on the log likelihood by removing dependencies between variables. For more details, about the variational inference algorithm, please refer to Blei *et al.* (2003).

Over the last two decades, a large number of topic modeling approaches have been introduced. Most of those approaches are based on the LDA topic modeling approach. A few of those models relaxing the statistical assumptions of LDA, e.g., relaxing the bag-of-words assumption, so that the order of words is incorporated by generating words conditioned on the previous word (Wallach (2006)), or relaxing the assumption that the order of documents within the corpus is irrelevant allows to account for topics changing over time (Blei and Lafferty (2006); Wang *et al.* (2008)). Teh *et al.* (2006) provide a solution for the need to know beforehand fixed and known number of topics in LDA by determining the number of topics during posterior inference. Other topic modeling approaches enhance LDA by incorporating structure from metadata, e.g., author-topic

model, introduced by Rosen-Zvi *et al.* (2004), or RTM, introduced by Chang and Blei (2009) extending LDA by considering links between documents.

2.2.2 Document-Topic Similarity

Generally, different topic modeling approaches are available. In this dissertation, we follow the idea of Blei *et al.* (2003) that a per-document topic probability distribution θ and a per-topic word probability distribution ϕ exist. We represent a topic model generated from documents in a corpus $\mathcal{D}$ by the tuple (θ, ϕ) denoted by $\mathcal{M}(\mathcal{D})$.

Definition 2.2.1 (Topic Model)**.** A topic model is a corpus-driven representation of document topic probability distribution θ and topic word probability distribution ϕ resulting directly from the documents in corpus $\mathcal{D}$, e.g., using the LDA approach from Section 2.2.1. Mathematically, we define a topic model $\mathcal{M}(\mathcal{D})$ as a tuple of the two probability distributions:

$$\mathcal{M}(\mathcal{D}) = (\theta, \phi). \tag{2.11}$$

Generally, having a corpus-driven topic model $\mathcal{M}(\mathcal{D})$, we can represent each document $d \in \mathcal{D}$ using the corpus-specific document topic probability distribution θ_d. Having the document topic probability distribution θ_d for each document $d \in \mathcal{D}$, we can estimate the similarity between two documents $d_i \in \mathcal{D}$ and $d_j \in \mathcal{D}$ by analyzing the similarity between the corresponding document topic probability distributions θ_{d_i} and θ_{d_j}.

The Hellinger distance, introduced by Hellinger (1909), is a metric for measuring the distance between two probability distributions. Thus, for each document $d \in \mathcal{D}$ we use the Hellinger distance to identify a set of documents from the same corpus each having similar document topic probability distribution to d. Given two documents d_i and d_j, we can represent the documents topic probability distributions by θ_{d_i} and θ_{d_j}, respectively. The Hellinger distance $H(\theta_{d_i}, \theta_{d_j})$ between both document topic probability distributions θ_{d_i} and θ_{d_j} is then given by:

$$H(\theta_{d_i}, \theta_{d_j}) = \frac{1}{\sqrt{2}} \sqrt{\sum_{k=1}^{K} \left(\sqrt{\theta_{d_i,k}} - \sqrt{\theta_{d_j,k}} \right)^2}, \tag{2.12}$$

where K refers to the number of topics that have been used within the topic model. The resulting Hellinger distance between the document topic probability distributions is a value between 0 and 1 because of the normalization term $\frac{1}{\sqrt{2}}$ in Expression (2.12). Sometimes factor $\frac{1}{\sqrt{2}}$ in front of the sum is omitted resulting in a Hellinger distance range from zero to the square root of two. Generally, the distance between the same document is zero, represented by $H(\theta_{d_i}, \theta_{d_i}) = 0$.

Having generated a topic model $\mathcal{M}(\mathcal{D})$ from all documents within a corpus $\mathcal{D}$, it is feasible to calculate the Hellinger distance between every combination of two documents

$d_i \in \mathcal{D}$ and $d_j \in \mathcal{D}$. Generally, the Hellinger distance between two documents d_i and d_j changes with the collection of documents in $\mathcal{D}$, because the underlying topic word probability distribution of the topic model depends on the words of all documents in corpus $\mathcal{D}$.

Example 2.2.1 (Hellinger distance)**.** Given a topic model $\mathcal{M}(\mathcal{D})$ generated from the documents in corpus $\mathcal{D}$ the Hellinger distance between two documents $d_i \in \mathcal{D}$ and $d_j \in \mathcal{D}$ results from the corresponding per-document topic probability distributions θ_{d_i} and θ_{d_i}.

Let us assume that we have $K = 5$ topics and the document topic probability distributions for d_i and d_j are given by:

$$\theta_{d_i} = (0.1, 0.17, 0.1, 0.6, 0.03) \tag{2.13}$$

$$\theta_{d_j} = (0.15, 0.05, 0.05, 0.7, 0.05) \tag{2.14}$$

Then, the Hellinger distance $H(\theta_{d_i}, \theta_{d_i})$ between both document topic distributions is given by 0.2224.

In addition to the Hellinger distance, there are further methods available to compare two probability distributions, e.g., the Kullback-Leibler divergence, which has been introduced by Kullback and Leibler (1951). However, the Kullback-Leibler divergence is not symmetric and the divergence depends on the *direction* comparing two distributions, e.g., the divergence of per-document topic distribution d_i and d_j is not the same as for d_j and d_i.

2.3 Corpus Annotation

Corpus annotation is an extensive field in corpus linguistic comprising different tasks, such as NER, POS tagging, or IE. We follow the definition of Leech (2005): Corpus annotation can be described as the task of adding interpretative linguistic data to documents in a corpus.

Generally, the granularity of annotations depends on the application and a single annotation may cover a word, a sentence, a paragraph, a document, or an entire corpus. In this section, we give an overview of linguistic annotations for corpora containing documents of written texts and describe two types of annotations, namely

(i) grammatical word tags, which represent for a word the particular part of speech, e.g., nouns, verbs, adjectives, adverbs, etc., and

(ii) semantic annotations, which represent for a text chunk of a document a concept, instance, or an individual from an ontology.

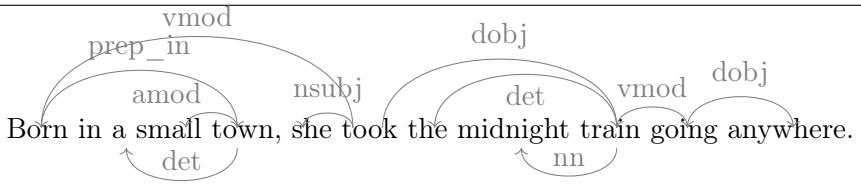


Figure 2.3: Dependency graph containing the grammatical word tagging of a text snippet.

Grammatical Word Tags Grammatical word tagging started in the early 1970s and it is the discipline of adding grammatical annotations to a corpus by specifying the grammatical characteristics of text in documents. The first corpus for grammatical word tagging is the Brown Corpus (Francis and Kucera, 1979). The tag set contains 77 word-class labels to identify not only major part of speech, such as nouns, verbs, or prepositions, but also subclasses like singular and plural nouns, or comparative and superlative adjectives. Francis and Kucera successfully tagged about 80% of words in the documents of the Brown Corpus to the correct classes. For the remaining 20% of words exist ambiguities and eliminating all remaining ambiguities was performed by manual editing the corpus. In the following years, further tagging projects followed, e.g., the LOB Corpus Marshall (1983). Next, we present an example of grammatical word tagging for a single sentence.

Example 2.3.1 (grammatical annotation)**.** In Figure 2.3, we illustrate an example for grammatical word tagging for the sentence: *Born in a small town, she took the midnight train going anywhere.*, where, e.g., **det** is a determiner, representing a relation between the head of a noun phrase (NP). **dobj** is a direct object, **nsubj** is a nominal subject, **prep_in** is a preposition, **vmod** is a verb modifier, **nn** is a noun, and **amod** is a adjectival modifier.

Grammatical annotations are important to identify the relationship between words to extract the semantics from a sentence. Generally, researchers look for automatic tagging methods to identify the class of words, since manually tagging words is impracticable. Different challenges exist making (automatic) grammatical annotations to an ongoing research field. Some well-known challenges are given by:

(i) Multiwords – At least two orthographic words correspond to one morphosyntactic word. A morphosyntactic word is an abstract word unit representing a word in terms of its grammatical properties.

(ii) Merged words – one orthographic word corresponds to many morphosyntactic word.

(iii) Phantom words – words are only implicit part of a given text, e.g., the short sentence *Looks good.* has the phantom word *it* before the word *Looks*. This means the phantom word extends the sentence resulting in *It looks good.*

(iv) Choice of tagset – select a tag-set and encoding for them to annotate words.

Next, we give an overview of semantic annotations.

Semantic annotations Generally, there is no single definition for semantic annotations. For Bechhofer *et al.* (2002) a semantic annotation consists of some rich and machine processable semantic information. Leech (2005) has described a semantic annotation as the process of adding information about the semantic category of words. Talantikite *et al.* (2009) have introduced semantic annotation as a description that is linked to an entity (from a text) and a semantic annotation is a referent to an ontology. Liao *et al.* (2011) describe that designing an appropriate ontology is the first step of the semantic annotation process.

Over the last decade, additional definitions for semantic annotations have been introduced. However, we focus on all definitions having in common that semantic annotation is the process of linking extractable entities from a text to an external source of data like an ontology representing a specification of a conceptualization.

Generally, semantic annotations add data to documents being *somehow* related to the text of documents, e.g., a semantic annotation describes an entity via references to concepts, instances, or individuals from an ontology, where concepts might be *people*, *places*, *organizations*, or *products*. Semantic annotation systems analyse the text of a document trying to identify entities and estimating relationships to other entities in an external source. In Example 2.3.2, we give a short example for enriching text with semantic annotations by mapping entities occurring in the text to externally available concepts.

Example 2.3.2 (Semantic annotation)**.** Let us assume, that the following sentence is in a document of a corpus and an external source is available containing concepts:

> *"Born in a small town, she took the midnight train going anywhere."*

Depending on the external data source, we can create different mappings between concepts and words. Having an external data source containing concepts `train` and `town`, it is possible to map the two words *train* and *town* to the concepts of the same name, respectively. Depending on the data source, an agent might map the following additional semantic annotation (`train instance-of transportation`) to `train`.

Enriched documents in a corpus with annotations results in an annotated corpus. The annotated corpus represents a more valuable resource than the original corpus, since different additional tasks can be performed on the annotated corpus. Unlike classic text annotations, which are often only for the reader's reference, semantic annotations can also be used for tasks like automatically find documents based on annotations or identify hidden relationships between entities in a text.

For the sentence in Example 2.3.2, an automatic annotation system might add additional semantic annotations, e.g., by mapping `she` to concepts like `person`, `mammal` or directly to an individual. Generally, there is no single *correct* semantic annotation for

a document and semantic annotations not necessarily describe the semantic meaning of a document, since semantic annotations depend on the subjective interpretation of a document that is influenced by the context, e.g., the documents in a corpus.

Thus, in this dissertation, we introduce subjective content descriptions and present different methods to enrich documents with subjective content descriptions while considering the content of a document and the corpus-driven context. Compared to semantic annotation systems, we focus on available content descriptions associated with documents in a corpus, e.g., to enrich only weakly annotated documents with additional subjective content descriptions from related documents from the same corpus.

In Part I, we introduce subjective content descriptions and present different approaches to enrich documents in a corpus with those descriptions.

Part I

Subjective Content Descriptions

The following workshop paper has introduced the idea of unsupervised text annotations for documents, providing first definitions, and algorithms in the domain of context-aware corpus annotation using subjective content descriptions:

> Tanya Braun, Felix Kuhr, and Ralf Möller. Unsupervised text annotations. In *Formal and Cognitive Reasoning - Workshop at the 40th Annual German Conference on AI (KI-2017)*, 2017

The initial idea was extended regarding corpus enrichment using unseen documents, annotation enrichment of documents in the same corpus, estimating hidden content descriptions within texts, and adaptation of annotations from one corpus to another corpus. The extensions has been published in the following literature:

> Felix Kuhr, Bjarne Witten, and Ralf Möller. Corpus-driven annotation enrichment. In *2019 IEEE 13th International Conference on Semantic Computing (ICSC)*, pages 138–141. IEEE, 2019

> Felix Kuhr, Tanya Braun, Magnus Bender, and Ralf Möller. To Extend or not to Extend? Context-specific Corpus Enrichment. In *Proceedings of AI 2019: Advances in Artificial Intelligence*, pages 357–368. Springer, 2019

> Felix Kuhr, Tanya Braun, Magnus Bender, and Ralf Möller. Augmenting and automating corpus enrichment. *Int. J. Semantic Comput.*, 14(2):173–197, 2020

> Felix Kuhr, Tanya Braun, and Ralf Möller. Augmenting and automating corpus enrichment. In *2020 IEEE 14th International Conference on Semantic Computing (ICSC)*, pages 69–76. IEEE, 2020

> Felix Kuhr, Magnus Bender, Tanya Braun, and Ralf Möller. Context-specific adaptation of subjective content descriptions. In *Proceedings of AI 2020: Advances in Artificial Intelligence*. Springer, 2020

> Magnus Bender, Tanya Braun, Marcel Gehrke, Felix Kuhr, Ralf Möller, and Simon Schiff. Identifying subjective content descriptions among text. In *IEEE 15th International Conference on Semantic Computing, ICSC 2021, Virtual, January 27-29, 2021*, pages 451–458, 2021

> Magnus Bender, Tanya Braun, Marcel Gehrke, Felix Kuhr, Ralf Möller, and Simon Schiff. Identifying subjective content descriptions among text. *will be published in: Int. J. Semantic Comput.*, 15(4), 2021

Felix Kuhr, Magnus Bender, Tanya Braun, and Ralf Möller. Context-specific adaptation of subjective content descriptions. In *IEEE 15th International Conference on Semantic Computing, ICSC 2021, Virtual, January 27-29, 2021*, pages 451–458, 2021

This part contains the first four contributions of the dissertation. Chapter 3 starts with the foundation of SCDs, where the SCDs act as annotations for documents. We define comparison techniques for SCDs and documents available in the same corpus. Chapter 4 contributes a context-specific enrichment approach of corpora with new documents based on the similarity values of SCDs. Chapter 5 describes a corpus-driven SCD enrichment approach of documents using SCDs that are associated with other documents in the same corpus. Chapter 6 describes an approach for handling documents where SCDs are interleaved within the text of a document, while Chapter 7 presents a domain adaptation approach to enrich documents from one corpus with SCDs associated with documents in another corpus. Finally, Chapter 8 concludes this part.

Chapter 3

Foundation of Subjective Content Descriptions

In this chapter, we present the foundation of subjective content descriptions. First, we introduce a corpus-driven SCD-word probability distribution we use in the subsequent chapters for different tasks an agent might perform on a corpus. Second, we describe how to select the most probably suited SCDs for documents based on the corpus-driven SCD-word probability distribution.

3.1 Subjective Content Descriptions

This section provides the theoretical foundation for SCDs we have introduced in Definition 2.1.6. We present an additional representation for each SCD by taking a vector of length V, where $V = |\mathcal{V}_\mathcal{D}|$ s.t. each vector entry refers to a word in the vocabulary $\mathcal{V}$ of all documents in corpus $\mathcal{D}$. The vector entry itself is a probability value describing how likely it is that a word occurs in an SCD window surrounding the position associated with the SCD, yielding an SCD-word probability distribution for each SCD associated with documents in $\mathcal{D}$. In other words, given a corpus containing documents associated with SCDs, we can correlate SCDs and words in a window around the position of SCDs from documents in the corpus resulting in SCD word frequency vectors, one vector for each SCD. We use a word frequency vector to represent each SCD instead of a bit vector, since SCDs are not exclusively associated with a single document in a corpus and might occur more than once. Algorithm 1 generates the SCD-word probability distribution for all m SCDs in the SCD set $T(\mathcal{D})$ from corpus $\mathcal{D}$.

The input of Alg. 1 is a corpus $\mathcal{D}$ containing a set of documents associated with SCDs. We start with an initialization of an empty matrix $\delta(\mathcal{D})$ and fill the matrix with zeros (line 4). Afterwards, we fill the SCD-word distribution matrix $\delta(\mathcal{D})$ based on the SCDs and words occurring in the documents of corpus $\mathcal{D}$ using a maximum-likelihood strategy such that we count for each SCD t the number of occurrences of each word w in the corresponding windows $win_{d,\rho}$ of all documents in $\mathcal{D}$. We weight the occurrences by the influence value of each word in a window (line 9). At the end of each outer loop iteration, the SCD-word probability distribution of the current SCD t is normalized to

Algorithm 1 Forming SCD-word probability distribution matrix $\delta(\mathcal{D})$

```
1: function buildMatrix(Corpus 𝒟)
2:     Input: Corpus 𝒟
3:     Output: SCD-word probability distribution matrix δ(𝒟)
4:     Initialize an m × V matrix δ(𝒟) with zeros
5:     for each d ∈ 𝒟 do
6:         for each t ∈ T(d) do
7:             for ρ of t do
8:                 for each w ∈ win_{d,ρ} do
9:                     δ(𝒟)[t][w] += I(w, win_{d,ρ})
10:    Normalize δ(𝒟)[t]
11:    return δ(𝒟)
```

yield a probability distribution for each SCD over the complete vocabulary (line 10). Finally, Alg. 1 returns the SCD-word distribution matrix $\delta(\mathcal{D})$.

Formally, normalization is given by:

$$v_{i,j} = \frac{v_{i,j}}{\sum_{k=1}^{V} v_{i,k}}, \tag{3.1}$$

where the denominator contains the sum of all influence values for words occurring in a corresponding SCD window surrounding an SCD t. For each entry in $\delta(\mathcal{D})$, we divide the corresponding influence value by the sum of all influence values we have previously calculated. Expression (3.2) represents the SCD-word probability distribution by an $m \times V$ matrix $\delta(\mathcal{D})$, with the SCD-word probability distribution vectors forming the rows of the matrix:

$$\delta(\mathcal{D}) = \begin{array}{c} \\ t_1 \\ t_2 \\ \vdots \\ t_m \end{array} \begin{array}{c} \begin{array}{ccccc} w_1 & w_2 & w_3 & \cdots & w_V \end{array} \\ \begin{pmatrix} v_{1,1} & v_{1,2} & v_{1,3} & \cdots & v_{1,V} \\ v_{2,1} & v_{2,2} & v_{2,3} & \cdots & v_{2,V} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ v_{m,1} & v_{m,2} & v_{m,3} & \cdots & v_{m,V} \end{pmatrix} \end{array} \tag{3.2}$$

We illustrate the behaviour of Alg. 1 using an example.

Example 3.1.1 (SCD-word probability distribution)**.** Let us assume that we have a corpus containing a document d and that d has an SCD window $win_{d,\rho}$ for SCD t_1 with

t_1 positioned at the center of the window (w_{15}, underlined). Expression (3.5) represents the influence values based on Expression (2.2) with:

$$n = s(win_{d,\rho}) - 1 = 6, \tag{3.3}$$

s.t. $k \in \{0, \ldots, 6\}$, and entry positions corresponding to positions in Expression (3.4). Expression (3.4) represents the SCD window $win_{d,\rho}$.

$$win_{d,\rho} = (w_{21}, w_4, w_8, \underline{w_{15}}, w_{16}, w_{23}, w_{42}) \tag{3.4}$$

$$(0.015625, 0.09375, 0.234375, \underline{0.3125}, 0.234375, 0.09375, 0.015625) \tag{3.5}$$

Based on the innermost loop of Alg. 1 (line 8-9), seven entries of $\delta(\mathcal{D})$ are updated, e.g., for w_{21} at position 0:

$$\delta(\mathcal{D})[t_1][w_{21}] \mathrel{+}= 0.015625$$

where $\delta(\mathcal{D})[t_1][w_{21}]$ refers to $v_{1,21}$, which is incremented by 0.015625. Algorithm 1 updates $\delta(\mathcal{D})$ for the remaining words and then continues with the next window. When Alg. 1 is finished with d_1, it moves on to the next document in $\mathcal{D}$, going through all windows. After iterating over all documents, Alg. 1 repeats going through all documents and their windows for the remaining SCDs.

The model behind an SCD-word probability distribution matrix $\delta(\mathcal{D})$ is generative as one could now choose $M \ll m$ SCDs and sample a new document based on the chosen SCDs. Given the generative nature, we are now interested in the Most Probably Suited Subjective Content Descriptions (MPSCDs) to have generated words of a new, previously unseen document.

Thus, in the next section we look at the problem of selecting those SCDs that are most probably suited given the words of a new document.

3.2 Most Probably Suited Subjective Content Descriptions

This section presents the problem of estimating the MPSCDs for new documents an agent might be interested in extending its corpus with. Generally, new documents are not associated with SCDs and without having thoroughly processed a new document, the question for the agent is: *Does that document have anything of value to add in the given context and the corpus-specific task of an agent?* The problem is a decision making problem: Should the agent extend its corpus with the new document or should it not?

Faced with a new document d' having no SCDs associated with, an agent can divide d' into M word sequences, each representing a window and select for each word sequence an SCD. Since each window contains a sequence of words an agent can generate for each window a word frequency vector $\delta(win_{d',\_})$ based on the words in the corresponding

window. The agent compares the word frequency vector of each window with the word frequency vector of each SCD associated with documents in the corpus. Then, the agent can compare the word frequency vector of a window with the word frequency vectors of all SCDs associated with documents in the corpus by using the cosine of the angle between two document vectors. The SCD where the corresponding word frequency vector has the smallest distance (highest cosine similarity) to the word frequency vector of the window is associated with a window. We denote the associated SCD as the MPSCD describing the content of the new document.We argue to divide a new document d' into M windows and select for each window the corresponding MPSCD resulting in a sequence of MPSCDs, where each MPSCD is associated with a location in the new document and each MPSCD contains a similarity value.

Mathematically, the MPSCD problem asks for the M most probable SCDs for a new document d' given the SCD-word probability distribution matrix $\delta(\mathcal{D})$ and the words in d'. As we do not model an influence of one SCD on the next and as we place the M windows next to each other in d', we can separately calculate the probability for each window and associated the SCDs with d':

$$\{\arg\max_{t \in T(\mathcal{D})} P(t|\delta(win_{d',\rho_1}), \delta(\mathcal{D})), \ldots, \arg\max_{t \in T(\mathcal{D})} P(t|\delta(win_{d',\rho_M}), \delta(\mathcal{D}))\}. \tag{3.6}$$

The intuition is as follows: If d' is a document from the same context as documents in $\mathcal{D}$ or a close variation of the documents in $\mathcal{D}$, then Expression (3.6) yields MPSCDs with high probabilities for each window. If d' is an unknown document, the resulting MPSCDs vary in their probability. If the vocabulary or word composition is very different, the probabilities are very low on average. The closer vocabulary and word composition of d' get to the characteristics of documents in $\mathcal{D}$, the higher the probabilities of MPSCDs.

Based on the statistics of the corpus, we approximate Expression (3.6) for an SCD t in a window $win_{d',\rho}$ by determining the SCD in $\delta(\mathcal{D})$ with the most similar distribution compared to a vector representation of $win_{d',\rho}$ using influence values. The setting is as follows: Given a new document d' and the SCD-word probability distribution matrix $\delta(\mathcal{D})$, we select the M MPSCDs for d'. Based on M, M windows $win_{d',\rho}$ lie over the text of d' with a window size of $\sigma = \frac{\#words(d')}{M}$ and positions ρ starting at $\frac{\sigma}{2}$ and incrementing by σ. For each $win_{d',\rho}$, the SCD is unknown at the start, i.e., $t = \perp$. As the words in $win_{d',\rho}$ have an influence value based on Expression (2.2), an agent can build a vector $\delta(win_{d',\rho})$ of length V. The entries $\delta(win_{d',\rho})[w]$ are set to 0 for each word $w \in \mathcal{V}$ not in $win_{d',\rho}$ and set to $I(w, win_{d',\rho})$ otherwise. Using cosine similarity, the SCD most similar to $\delta(win_{d',\rho})$ is given by the cosine between both vectors:

$$\arg\max_{i \in T(\mathcal{D})} \frac{\delta(\mathcal{D})[i] \cdot \delta(win_{d',\rho})}{|\delta(\mathcal{D})[i]| \cdot |\delta(win_{d',\rho})|}. \tag{3.7}$$

Algorithm 2 estimates MPSCDs for d' using $\delta(\mathcal{D})$ given M. The output of the algorithm is a set of triples containing MPSCDs, the corresponding MPSCD similarity value, as

Algorithm 2 Selecting MPSCDs based on Similarities

1: **function** SELECTMPSCD(new document d', Number of SCDs M, matrix $\delta(\mathcal{D})$)
2: **Input**: new document d', Number of SCDs M, matrix $\delta(\mathcal{D})$
3: **Output**: Sequence of triples $\mathcal{W}_{d'}$
4: $\sigma \leftarrow \frac{\#words(d')}{M}$, $\mathcal{W} \leftarrow \emptyset$
5: **for** $\rho \leftarrow \frac{\sigma}{2}$; $\rho \leq \#words(d)$; $\rho = \rho + \sigma$ **do**
6: Set up a window $win_{d',\rho}$ of size σ around ρ
7: $\delta(win_{d',\rho}) \leftarrow$ new zero-vector of length V
8: **for** $w \in win_{d',\rho}$ **do**
9: $\delta(win_{d',\rho})[w] += I(w, win_{d',\rho})$
10: $t \leftarrow \arg\max_i \frac{\delta(\mathcal{D})[i] \cdot \delta(win_{d',\rho})}{|\delta(\mathcal{D})[i]| \cdot |\delta(win_{d',\rho})|}$ in $win_{d',\rho}$ ▷ Cosine similarity
11: $sim \leftarrow \max_i \frac{\delta(\mathcal{D})[i] \cdot \delta(win_{d',\rho})}{|\delta(\mathcal{D})[i]| \cdot |\delta(win_{d',\rho})|}$
12: $\mathcal{W} \leftarrow \mathcal{W} \cup \{(t, sim, win_{d',\rho})\}$
13: **return** $\mathcal{W}_{d'}$

well as the window-details. The outer loop (line 5 to line 12) iterates over the positions of the M SCDs, setting up a window $win_{d',\rho}$ and a vector representation $\delta(win_{d',\rho})$. Then, Alg. 2 calculates cosine similarities between $\delta(win_{d',\rho})$ and the SCD vectors in $\delta(\mathcal{D})$ based on Expression (3.7). The algorithm returns the SCD with the highest similarity value as MPSCD t for the window $win_{d',\rho}$. The approach rests on the following proposition:

Proposition 3.2.1. *For a new document d' Algorithm 2 estimates the (locally) M MPSCDs, i.e., for each window Expression* (3.7) *estimates SCDs of Expression* (3.6)*.*

We argue that the similarity between the influence distribution over the words in a window and the SCD-word probability distribution indicates that the SCD is most likely to generate the words in the window. Another SCD generating other words with high probability would not generate the words in the window with a high probability and as such, does not lead to a high similarity. The MPSCDs represent a local optimum based on the current setting of the windows.

The next chapter describes the enrichment of a given corpus with new documents using the corpus-specific SCD-word probability distribution matrix $\delta(\mathcal{D})$ and MPSCDs for all new documents resulting from the SCD-word distribution of the corpus. If an agent decides on adding a new document to the corpus and using the MPSCDs for additional corpus-specific tasks like query answering or document retrieval, optimizing the initial SCDs of d' might lead to more attuned SCDs. We describe the optimization of an initial set of SCDs associated with a document in Chapter 5.

Chapter 4

Context-specific Corpus Enrichment

This chapter presents an approach for extending a corpus with new documents and automatically enriching the new documents with SCDs while considering the context represented by the documents in a corpus. Aforementioned, documents in a corpus play an important role for an agent's task, and we assume that SCDs associated with documents are optimized for the task of the agent. An agent might be interested in extending its corpus with a new document only if the content of that document is relevant for its task, e.g., the new document contains content related to the content of other documents in the corpus. But, what should an agent do while presented with a new document, which typically has no associated SCDs? The question for the agent is: Does the content of a new document provide anything of value to add in the context of a given corpus? An agent might be interested in new documents being similar to documents already in the corpus or containing content extending the content of documents in the corpus. Generally, an agent can classify new documents into one category from a set of predefined categories $\mathcal{C}$, where each category represents the relationship between the content of new documents and the content of documents in the corpus. We use the following document categories an agent might be interested in:

(i) *sim* - new document is similar to at least one document in the corpus,

(ii) *ext* - new document is an extension of a document in the corpus,

(iii) *rev* - new document is a revision of a document in the corpus, and

(iv) *unrel* - new document is unrelated to all documents in the corpus.

Additionally, we argue that a new document provides a value for the task of an agent if the document is classified into category *sim*, *ext*, or *rev*, since a document classified to one of those three categories contains content related to content of documents in the corpus. We denote the process of extending a corpus with documents adding a value for the task of an agent with the notion *corpus enrichment*. Generally, the specific categories an agent is interested in is task-specific and depending on the task further categories might be beneficial for the agent.

We assume that the relatedness of a window's content to the content of documents in the corpus has an impact on the relatedness of the content in the next window. Different approaches exist to model a sequential impact. We use Hidden Markov Models modeling the sequential impact and use the relatedness of the content (related/unrelated) as hidden states and MPSCD similarity values as observations.

To estimate the category of a new document, we perform the following steps. First, we learn for each of the four document categories a corpus-specific HMM where each slice in an HMM concerns a window in a document and the hidden state in the HMMs concerns whether the SCD in the window is related or unrelated to the agent's task. Second, given a new document, we estimate for each window an MPSCD and use the corresponding sequence of MPSCD similarity values as observable input data for the Viterbi algorithm, introduced by Viterbi (1967), to estimate the most likely sequence of related/unrelated content (hidden states) for each of the four previously learned HMMs. This technique results in four probability values, one probability value for each of the estimated most likely sequences of hidden states. The probability values can be used to rank the most likely sequences. The category of a new document is given by the HMM resulting in the sequence with highest probability value.

Additionally, an agent can use the most likely sequence of hidden states to provide location-specific information about new data within the context of the specific corpus. If an agent decides to extend its corpus with new documents based on the documents' category it can even choose to retain the initially estimated MPSCDs, possibly adapting them with new SCDs, and use the SCDs as a basis for enriching SCDs in the corpus, in an automatic fashion.

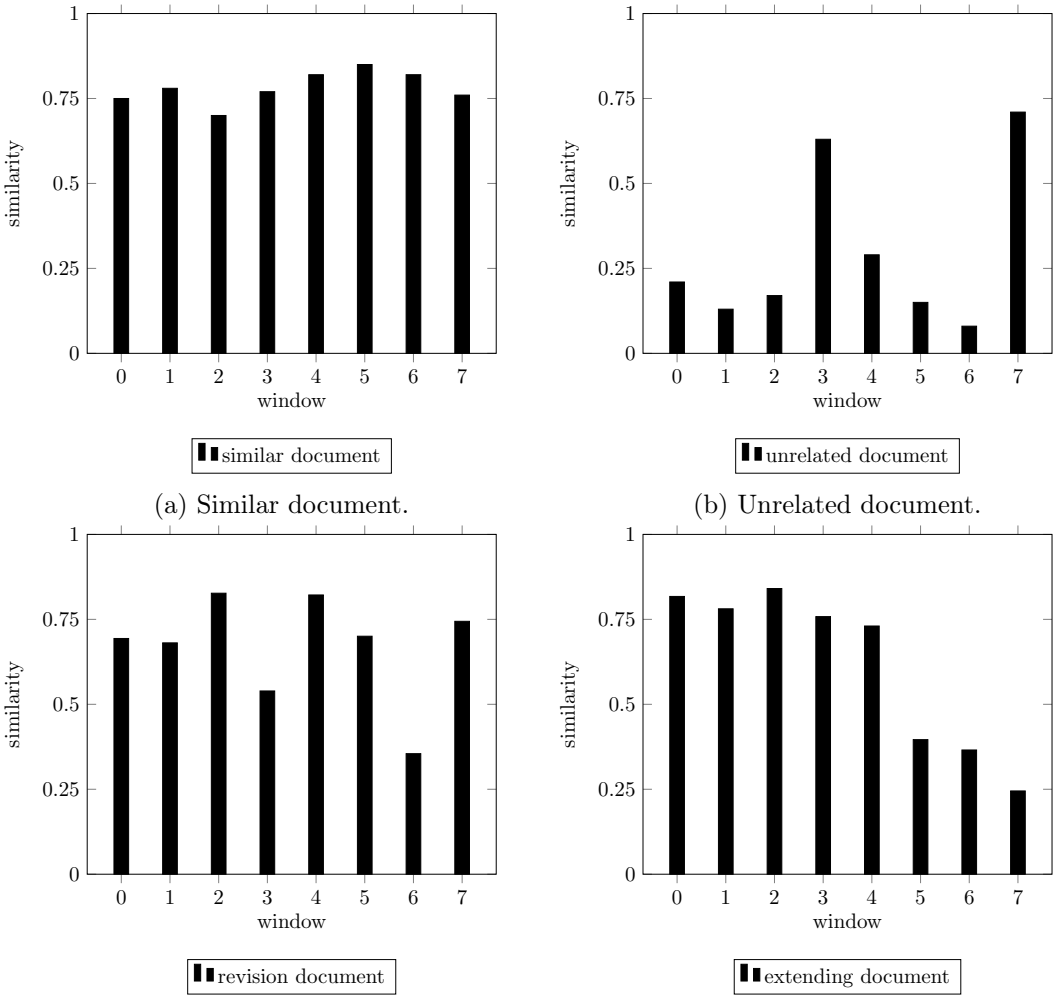
We introduce the problem of classifying a new document d' into a predefined category given an individual composition of documents in a corpus $\mathcal{D}$, and present an HMM-based approach to identify the category of new document d' by estimating the probability of the most likely sequence of hidden states from the observable MPSCD similarity values in d'. We then provide a decision making procedure and describe how an agent can use the most likely sequence to augment corpus enrichment with providing location-specific information about new data within the context of the specific corpus. Even though one might use any identified category relevant to a specific task, we consider the four document categories *sim*, *ext*, *rev*, and *unrel*.

Before going into details about document classification techniques, in the next section we first describe each document category and show how document classification has a chance at success using the sequence of MPSCD similarity values.

4.1 Document Categories and Similarity Values

We assume that a document is labeled with one of the four previously introduced document categories and assume that the category of a document can be estimated by

considering the MPSCD similarity values. Before presenting details about the document categories, let us consider an example for document categories.



(a) Similar document.

(b) Unrelated document.

(c) Revision document.

(d) Extending document.

Figure 4.1: Representation of MPSCD similarity value characteristics for a similar (a), unrelated (b), revision (c), and extending (d) document in an exemplary way.

Example 4.1.1 (document category)**.** Assume that an agent is presented with four documents of similar length and each of them from one of the four categories *sim*, *ext*, *rev*, and *unrel*, respectively. Figure 4.1 presents the similarity value characteristics of the MPSCDs for each of the four documents using seven windows to segment the content within the documents. The x-axis represents the SCD window number within each docu-

ment, and the values on the y-axis represent the similarity value of MPSCDs. Generally, the similarity values change slightly between neighboring windows for both similar and unrelated documents and only few major shifts between neighboring windows are given. But the MPSCD values of similar documents are considerably higher compared to unrelated documents. Extending documents are characterized by high similarity values in the initiating SCD windows and smaller similarity values in the later windows. The similarity value for the windows' MPSCD of revision documents behave like similar documents. However, there are windows having a significant change in the similarity value of the neighboring windows, namely, those windows having new content which is unrelated in the corpus-specific context.

Based on the observations shown in the last example, we describe the document categories in more detail.

Similar documents (*sim*). The content of a new document d' is classified as category *sim* if the document's content is related to the content of a subset of documents in corpus $\mathcal{D}$, i.e., the new document tells about the same event, same person, or same domain yielding to high MPSCD similarity values using Alg. 2. Thus, values in the MPSCD similarity sequence are mostly high and contain only few entries with slightly lower values. In other words, the range of MPSCD similarity values is small. A new document d' is a similar document w.r.t. the documents in a corpus $\mathcal{D}$ if

$$\forall win_{d',\rho} \in d' : \max_i \frac{\delta(\mathcal{D})[i] \cdot \delta(win_{d',\rho})}{|\delta(\mathcal{D})[i]| \cdot |\delta(win_{d',\rho})|} > th_1. \tag{4.1}$$

Threshold th_1 describes the minimum required relatedness for each window in d'. The threshold is subjective and depends on the individual composition of documents in $\mathcal{D}$.

Unrelated documents (*unrel*). The content of a new document d' is classified as category *unrel* if the document's content is unrelated to the content of all documents in corpus $\mathcal{D}$. The values in the MPSCD similarity sequence of d' are mostly low and only a small number of windows contain higher similarity values. Thus, we define a new document d' as an unrelated document w.r.t. the documents in a corpus $\mathcal{D}$ if

$$\forall win_{d',\rho} \in d' : \max_i \frac{\delta(\mathcal{D})[i] \cdot \delta(win_{d',\rho})}{|\delta(\mathcal{D})[i]| \cdot |\delta(win_{d',\rho})|} < th_2. \tag{4.2}$$

Threshold th_2 describes the maximum relatedness for each window in document d'. The threshold is subjective and depends on the individual composition of documents in $\mathcal{D}$.

Extending documents (*ext*). The content of a new document d', representing an extension of another document $d \in \mathcal{D}$ is mostly related to the content of at least one document

in $\mathcal{D}$ but additionally contains content which is not available in documents of $\mathcal{D}$, i.e., the new document is an extended version of another document in corpus $\mathcal{D}$. Thus, new document d' can be represented by $d' = [a, a, ..., b, b, ...]$, where each a represents one of the first windows of d' containing MPSCD similarity values greater than th_1 and b represents the second part of windows having MPSCD similarity values smaller than th_2. We can represent the windows properties using the following regular expression: a^+b^+.

Revision documents (rev). The content of a new document d', representing a revision of another document $d \in \mathcal{D}$ is generated by *appending*, *replacing*, or *removing* some sentences of document d. The MPSCD similarity value of most windows is greater than th_1 and the MPSCD similarity value of few windows is less than th_2. We can represent the described behaviour as the following regular expression: $(a^+b \mid b^+a)(a \mid b)^*$. Again, a and b represent MPSCD similarity values greater than th_1 and smaller than th_2, respectively.

Next, we define the problem of classifying a new document into one of the four document categories and present an approach to solve the problem using four HMMs, one for each document category.

4.2 Document Classification Problem

After estimating the MPSCDs for a new document d' using Alg. 2, we can extract a sequence of MPSCD similarity values over the SCD windows in d'. The document classification problem asks for the most probable category of document d' given the observable sequence of MPSCD similarity values and documents in $\mathcal{D}$. More technically, we can represent the classification problem by Expression (4.3).

$$\arg\max_{c \in \mathcal{C}} P(c|\mathcal{W}_{d'}), \tag{4.3}$$

where $\mathcal{W}_{d'}$ represents the document-specific triple containing SCDs, corresponding similarity values and windows. As described in the previous section, we follow the idea that if the content in a window of a new document d' is related to the agent's task, then d' contains high similarity values, which may vary over the course of a new document, with related and unrelated parts mixing. Generally, we can only look at the observable similarity values of MPSCDs but cannot directly observe if the content in a window is related to the task of an agent. The consequence of this consideration is a sequence of hidden states, encoding the relatedness of the document's content, and a sequence of observations given by the similarity values of MPSCDs. HMMs can model the described setup and we are interested in solving the document classification problem by learning four HMMs, one for each category of documents, determining the most likely sequence of hidden states in the new document d'. Finally, we return the category associated with the HMM having the highest probability of hidden state sequence. Calculating the most

likely sequence of hidden states allows us to analyse documents "over time", i.e., sentence by sentence. Additionally, we can consider the sequence of the MPSCD similarity values, instead of simply using statistic values like the maximum or minimum in a set of MPSCD similarity values.

In Section 4.6, we evaluate the performance of an HMM-based approach against a multi-dimensional decision approach, where the multi-dimensional decision approach considers different properties of the sequence of MPSCD similarity values.

4.3 Category-specific HMMs

We generate for each category an HMM to detect the category of a new document.

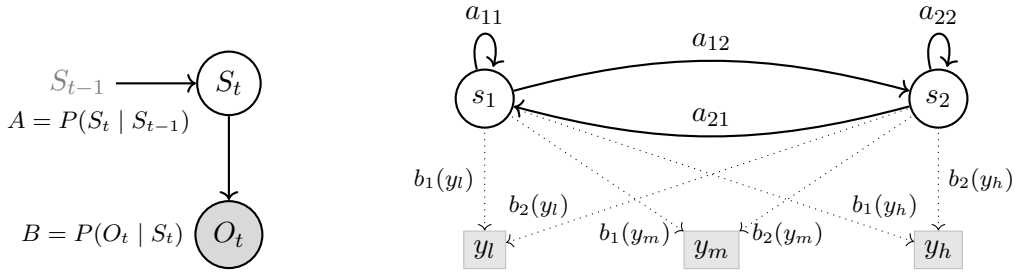
Definition 4.3.1 (Hidden Markov model)**.** An HMM $\lambda = (\Omega, \Delta, A, B, \pi)$ for classifying documents of some category consists of

- a set of hidden states given by $\Omega = \{s_1, ..., s_n\}$, where $n = 2$, with state s_1 representing related content and s_2 representing unrelated content,
- an observation alphabet $\Delta = \{y_1, \ldots, y_m\}$, where each observation symbol represents a range of MPSCD similarity values,
- a transition probability matrix A representing the probability between all possible state transitions $a_{i,j}$ of the two hidden states $s_1, s_2 \in \Omega$,
- an emission probability matrix B representing the probability to emit a symbol from observation alphabet Δ for each hidden state in Ω, and
- an initial state distribution vector $\pi = \pi_0$.

Example 4.3.1 (Graphical Representations)**.** Figure 4.2 contains two graphical representations of an HMM $\lambda = (\Omega, \Delta, A, B, \pi)$ with observation symbols $\{y_l, y_m, y_h\} \in \Delta$ and two states $\{s_1, s_2\} \in \Omega$. Figure 4.2a shows a one-time slice representation of the HMM. Figure 4.2b shows the HMM as the corresponding state-transition system, where observable variables are denoted in grey.

Definition 4.3.2 (Transition Probability)**.** With $\sum_{j=1}^{n} a_{i,j} = 1$, for all i, the entries of transition probability matrix A between states $s_i, s_j \in \Omega$, are given by

$$a_{i,j} = P(s_j|s_i), \tag{4.4}$$



(a) One-time slice graph (b) State-transition system

Figure 4.2: Hidden Markov model with hidden variable S_t with two possible states $\{s_1, s_2\}$ emitting an observation O_t with three possible values $\{y_l, y_m, y_h\}$.

and represented by the following state transition matrix A:

$$A = \begin{array}{c} \\ s_1 \\ s_2 \\ \vdots \\ s_n \end{array} \begin{array}{c} \begin{array}{cccc} s_1 & s_2 & \cdots & s_n \end{array} \\ \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{n,1} & a_{n,2} & \cdots & a_{n,n} \end{pmatrix} \end{array}$$

Definition 4.3.3 (Emission Probability)**.** The entries of emission probability matrix B represent the probability to emit symbol $y_k \in \Delta$ in hidden state $s_j \in \Omega$ and, with $\sum_{j=1}^{m} b_j(y_k) = 1$, are given by

$$b_j(y_k) = P(y_k|s_j). \tag{4.5}$$

The semantics of λ is given by unrolling λ for a given number of slices and building a full joint distribution. For a set of document categories $\mathcal{C}$, $\mathcal{H}$ contains an HMM for each document category in $\mathcal{C}$. All $\lambda \in \mathcal{H}$ have the same hidden states Ω and observation vocabulary Δ but different transition probabilities and emission probabilities. We represent the emission probabilities between observations and hidden states using the following observation matrix B:

$$B = \begin{array}{c} \\ s_1 \\ s_2 \\ \vdots \\ s_n \end{array} \begin{array}{c} \begin{array}{cccc} y_1 & y_2 & \cdots & y_m \end{array} \\ \begin{pmatrix} b_1(y_1) & b_1(y_2) & \cdots & b_1(y_m) \\ b_2(y_1) & b_2(y_2) & \cdots & b_2(y_m) \\ \vdots & \vdots & \vdots & \vdots \\ b_n(y_1) & b_n(y_2) & \cdots & b_n(y_m) \end{pmatrix} \end{array}$$

Generally, for a corpus $\mathcal{D}$, both, the transition probability matrix A and the emission probability matrix B for each document category specific HMM is unknown. One famous

technique for learning both, the transition probability matrix and the emission probability matrix of an HMM is the Baum-Welch algorithm, introduced by Baum *et al.* (1970), which is a special case of the well-known EM algorithm (Dempster *et al.* (1977)). Using a set of documents where each document is from a specific category $c \in \mathcal{C}$, one can calculate for a hold-out set the MPSCDs and the corresponding MPSCD similarity values s.t. we can train one HMM for each category on the data using the Baum-Welch algorithm.

The observation alphabet Δ requires discretizing similarity values. As a function $f : [0, 1] \mapsto \Delta$, it maps a similarity value x to one of the m symbols in Δ:

$$f(x) = \begin{cases} y_1 & 0 \leq x < th_1 \\ y_2 & th_1 \leq x < th_2 \\ \vdots & \\ y_m & th_{m-1} \leq x < 1 \end{cases} \tag{4.6}$$

Generally, the discretization and its thresholds depend on the task of an agent and can be adapted to each problem individually. In our case study, we use the following setting: $m = 3$, $th_1 = 0.3$ and $th_2 = 0.7$ such that $f(x)$ maps MPSCD values to y_l, y_m, and y_h, referring to low, medium, and high values.

Next, we present an approach for estimating the most probable category of a new document by providing the sequence of observable MPSCD similarity values to the corpus- and category-specific HMMs.

4.4 Detecting the Category of a New Document

To solve the document classification problem, an agent has to identify the most likely sequence of hidden states from alphabet Ω, given a sequence of observation symbols from alphabet Δ, in each HMM of the category-specific HMMs. Each of the most likely sequences is associated with a probability value. The document category is given by the category for which the HMM with the highest probability for its sequence has been learned.

The task of determining which sequence of variables is the underlying source of some sequence of observations is called a *decoding task*. We calculate for each HMM the most likely sequence of hidden states using the Viterbi algorithm, which makes use of the dynamic programming trellis for computing the most likely hidden state sequence S for an observation sequence O. Before presenting a complete specification of the document classification approach, let us consider an example of a most likely sequence given a set of MPSCD similarity values.

Example 4.4.1 (Most likely sequence)**.** Let us assume that we have calculated a set of MPSCDs for a new document d' by using Alg. 2 and the corresponding MPSCD similarity

Algorithm 3 Classifying Documents

1: **function** DOCCATEGORYDETECTION($\mathcal{W}_{d'}$, $\mathcal{H}$, f)
2: **Input**: Seq. of triples $\mathcal{W}_{d'}$: SCD (t), similarity value (sim), and window ($win_{d',\rho}$), HMM set $\mathcal{H}$, discretization function f
3: **Output**: tuple s containing most likely sequence S and the HMM λ
4: $p \leftarrow 0$ ▷ current highest probability
5: $s \leftarrow$ initialize ▷ output tuple (S, λ)
6: $O \leftarrow$ initialize ▷ observation sequence
7: **for** each $(t, sim, win_{d',\rho}) \in \mathcal{W}_{d'}$ **do**
8: $O \leftarrow O \circ f(sim)$ ▷ discretize
9: **for** each $\lambda \in \mathcal{H}$ **do**
10: $S \leftarrow$ VITERBI(λ, O) ▷ most likely hidden state sequence S
11: $y \leftarrow prob(S)$ ▷ probability of S
12: **if** $y > p$ **then**
13: $p \leftarrow y$
14: $s \leftarrow (S, \lambda)$
15: **return** s

values are as follows:

$$\mathcal{W}_d = (0.29, 0.41, 0.59, 0.48)$$

Applying function $f(x)$ on all four values leads to the following observation sequence:

$$O = (y_l, y_m, y_m, y_m)$$

Assume that the hidden state sequence for a specific configuration of the transition and emission probability matrices of an HMM is given by

$$S = (s_2, s_1, s_1, s_1).$$

Figure 4.3 represents the trellis of the observation sequence O, where the thick arrows indicate the most probable transitions between the hidden states and the dotted lines represent all possible hidden state transitions.

Algorithm 3 describes how to estimate the most probable category of a new document using the category-specific HMMs. The input parameters are given by the MPSCD similarity values in $\mathcal{W}_{d'}$ (output of Alg. 2), the category-specific HMMs $\mathcal{H}$, and function f. In line 4 to line 6, Alg. 3 initializes a temporary variable p, the output tuple s, and discretized observation sequence O. In line 7 to line 8, Alg. 3 calculates the discretized observation sequence from MPSCD similarity values of d' using function f.

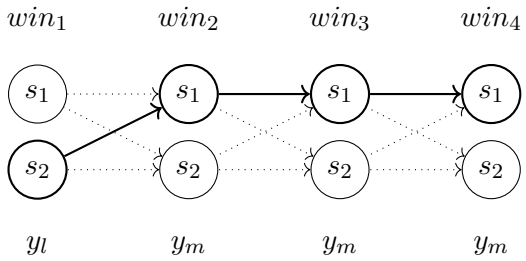


Figure 4.3: Trellis of $O = (y_l, y_m, y_m, y_m)$, leading to the following most likely sequence of hidden states: (s_2, s_1, s_1, s_1).

Afterwards, Alg. 3 iterates over the given HMMs in $\mathcal{H}$. In each iteration, the algorithm calculates a most probable sequence S given observables O using the Viterbi algorithm (line 10). In line 12 to line 14, Alg. 3 tests if the probability $prob(S)$ of the current most likely sequence S is higher than the previously seen highest probability. If the probability is higher, Alg. 3 saves the current most likely sequence S and the HMM λ as a tuple (S, λ) in variable s.

After iterating over all HMMs in $\mathcal{H}$, Alg. 3 terminates and returns the most likely sequence s that has exhibited the highest probability among all most likely sequences as well as the corresponding HMM. We then use this output as a basis for the decision regarding corpus enrichment. Before moving on to decision making, let us consider an example for how document categories and sequences may interact.

Example 4.4.2. Assume that we have an unrelated document with similarity values that have led to the following observation sequence as well as the most likely sequence as output of Alg. 3, meaning the HMM learned on unrelated documents has produced the most likely sequence with the highest probability:

$$O_{unrel} = (y_l, y_m, y_l, y_l) \tag{4.7}$$

$$S_{unrel} = (s_2, s_2, s_2, s_2) \tag{4.8}$$

With observations of y_l mainly, the most likely sequences of the other HMMs have a much lower probability as the evidence for unrelated content is very high, which is associated with low probabilities in them. Given, e.g., an extension of a document, the observation sequence may look as follows with the HMM learned on extensions yielding the most likely sequence with highest probability also given:

$$O_{ext} = (y_h, y_m, y_l, y_l) \tag{4.9}$$

$$S_{ext} = (s_1, s_1, s_2, s_2) \tag{4.10}$$

The HMM trained on unrelated documents can only explain the last part with high probability whereas the HMM trained on extensions can explain both parts with high probability.

Estimating the category of a new document d' can help an agent making a decision on extending a corpus with the new document d' or not, e.g., if an agent is interested in new documents containing similar content to documents in the corpus the agent can simply add new documents classified as similar document.

4.4.1 Automatic Decision Making: Corpus Enrichment

Given corpus $\mathcal{D}$ and the corresponding set of SCDs in $T(\mathcal{D})$, an agent has to perform following steps *offline*:

(i) Build an SCD-word distribution matrix $\delta(\mathcal{D})$ based on the SCDs in $T(\mathcal{D})$.

(ii) Train the category-specific HMMs in $\mathcal{H}$ using the documents in corpus $\mathcal{D}$ and the corresponding SCD-word probability distribution matrix $\delta(\mathcal{D})$.

(iii) Specify a function f for discretizing similarity values to build an observation alphabet Δ.

(iv) Specify the document category an agent is interested in.

Relevant document category may be extensions of documents for an agent to extend its own corpus. If a corpus is already very large, an agent may be interested in updating its documents with newest revisions of the documents. If a task changes and documents no longer appear to be helpful in solving its task, unrelated documents may provide the most added value (without any further information about the agent's information need). As just indicated, specifications may even change over time when a corpus grows and a need for specific documents becomes apparent.

Faced with a new document d', an agent performs the following steps *online*:

(i) Use Alg. 2 to estimate the MPSCDs similarity values for document d'.

(ii) Use the output of Alg. 3 to determine the category of d'.

(iii) Based on document's category and the document category an agent is interested in, add or forgo d'.

Basing the decision on most likely sequences enables an agent to further process the most likely sequence with the highest probability and analyse documents "over time", i.e., sentence by sentence. As such, this sequence allows for augmenting corpus enrichment by being able to pinpoint interesting positions in a document, which we consider next.

4.4.2 Augmenting Enrichment: Positions of Interest

Let us assume that an agent is interested in extending a corpus $\mathcal{D}$ with new documents and that the documents are classified as unrelated documents but share some small parts containing related content with at least one document $d \in \mathcal{D}$. The output of Alg. 3 is a tuple containing the most likely hidden state sequence S and the document category λ represented by most likely HMM. Sequence S represents for each window in d' if the content is related or unrelated to content of documents in $\mathcal{D}$. We can use the sequence S to extract information about the content similarity in an SCD window instead of only having information about the similarity values of the most probably suited SCD relating to a specific window. If a new document d' is unrelated to other documents in $\mathcal{D}$ but parts of d' contain content related to documents in $\mathcal{D}$, then the agent might be interested in the parts containing related content to identify if document d' might contain new content relating to the corpus-specific context, even if d' is classified as unrelated. We describe these parts in a new document d' as Positions of Interests (PoIs).

Identifying PoIs in a new document d' requires detecting the category of d'. Depending on the category of d', an agent might be interested in different positions within document d'. Thus, we specify the following PoIs depending on the document's category as follows:

(i) For documents in category *sim*, sections containing unrelated content represent PoIs, s.t. an agent can analyse the unrelated content.

(ii) Documents in category *ext* contain related content in the beginning of the document which changes to unrelated content when the document extension starts. The position where the similarity values changes represents the PoI in *ext*.

(iii) For documents of category *rev*, the PoI is given by those positions containing modified (unrelated) content, and

(iv) for documents in category *unrel*, the sections containing related content represent PoIs, s.t. an agent can check if the unrelated document adds a value for the corpus, e.g., by analyzing the queries already performed on the document retrieval system.

4.5 Optimizing Subjective Content Descriptions

In the last section, we have introduced an approach for classifying new documents into one of four categories. The introduced document classification approach is based on an initial set of SCDs associated with documents of corpus $\mathcal{D}$ as well as the estimated MPSCDs and their corresponding similarity values for the windows of the new document. In this section, we present two additional approaches for optimizing the quality of initially estimated MPSCDs for a new document d' an agent is faced with. The first approach adapts the size and position of initial SCD windows. The second approach reduces the

documents in the initial corpus based on the document category an agent is interested in resulting in an new SCD-word probability distribution $\delta(C_{d'})$, where $C_{d'}$ represents the reduced corpus.

Adapting Subjective Content Description Windows

In Section 3.2, we have introduced Alg. 2 returning $\mathcal{W}_d$ containing a set of triples each containing a located MPSCD, the MPSCD similarity value as well as the corresponding window. Both, the initial located MPSCDs and the corresponding similarity values are based on SCD-word probability distribution $\delta(\mathcal{D})$ and the position of the windows. We argue that the MPSCDs associated with d' represent only a first estimate for d' because the position and size of the corresponding SCD windows might be non-optimal. If an agent is interested in extending its corpus with a document, then the agent benefits from optimized MPSCDs for d', e.g., to increase the document retrieval performance.

Thus, we aim to optimize the MPSCDs associated with d' by iteratively adjusting the size and position of all windows in a way that the overall similarity values of the M MPSCDs is at maximum. Algorithm 4 (see next page) presents the iterative MPSCD optimization approach improving the initial MPSCD similarity values for document d', such that the overall MPSCD-similarity value increases.

Algorithm 4 requires the following input parameters: (i) new document d', (ii) number M of windows, (iii) initial set of MPSCDs, corresponding cosine similarity values and window details ($\mathcal{W}_{d'}$), and (iv) SCD-word probability distribution $\delta(\mathcal{D})$. Finally, Alg. 4 returns $\mathcal{W}_{d'}$ containing the optimized triple of located SCDs, similarity values and windows for d'.

In line 5, Alg. 4 calculates the overall cosine similarity of the initial MPSCDs associated with d' using the similarities stored in $\mathcal{W}_{d'}$. Afterwards, in line 6 to line 19, Alg. 4 iteratively adjusts SCD windows trying to increase the cosine similarity between the adjusted influence vector and the SCD vector in $\delta(\mathcal{C}_{d'})$. The window adjustment is performed until a local optimum is reached.The algorithm adjusts each window in $\mathcal{W}_{d'}$ in the following four different directions without overlapping boundaries: (i) extend left boundary of the SCD-window to the left side (l^+), (ii) extend right boundary of SCD-window to the right side (r^+), (iii) shift left boundary of SCD-window to the right side (l^-), and (iv) shift right boundary of SCD-window to the left side (r^-). The first two window adjustments extend the size of $win_{d',\rho}$, while the last two window adjustments decrease the size of $win_{d',\rho}$. We calculate for each possible window adjustment the MPSCDs and choose the adjustment, for which the overall similarity is maximized.

Proposition 4.5.1. *For a corpus-extending document d' Algorithm 4 estimates the M (locally) most probably suited located SCDs and finally terminates.*

Algorithm 4 optimizes the initial M SCDs in $\mathcal{W}_{d'}$ by iteratively adjusting the M windows in a fashion s.t. the overall similarity between the influence vectors of all SCD

Algorithm 4 Optimizing MPSCDs

function OPTIMIZINGMPSCDS(d', M, $\delta(\mathcal{D})$, $\mathcal{W}_{d'}$)
  Input: document d', number of SCDs M, SCD-word probability $\delta(\mathcal{D})$, seq. of triples $\mathcal{W}_{d'}$
  Define: t^*, sim', sim^*, ρ, $winT_{d',\rho'}, sum, sum'$
  Output: $\mathcal{W}_{(d')}$
  $sum \leftarrow \sum_{(\_,sim,\_)\in\mathcal{W}_{d'}} sim$
  $sum' \leftarrow sum$
  while $sum' \geq sum$ **do**
    $sum \leftarrow sum'$
    for each $(t, sim, win_{d',\rho}) \in \mathcal{W}_{d'}$ **do**
      $sim^* \leftarrow sim$
      for each window adjustment $a \in \{l^+, r^+, l^-, r^-\}$ **do**
        $winT_{d',\rho'} \leftarrow win_{d',\rho}$
        Update $winT_{d',\rho'}$ based on a
        $sim' \leftarrow \max_i \frac{\delta(\mathcal{C}_{d'})[i]\cdot\delta(winT_{d',\rho'})}{|\delta(\mathcal{C}_{d'})[i]|\cdot|\delta(winT_{d',\rho'})|}$
        if $sim' > sim^*$ **then**
          $t^* \leftarrow \arg\max_i \frac{\delta(\mathcal{C}_{d'})[i]\cdot\delta(winT_{d',\rho'})}{|\delta(\mathcal{C}_{d'})[i]|\cdot|\delta(winT_{d',\rho'})|}$
          Update $(t, sim, win_{d',\rho}) \in \mathcal{W}_{d'}$ with $t = t^*$ and $win_{d',\rho} = win_{d',\rho'}$
          $sim^* \leftarrow sim', sim \leftarrow sim'$
    $sum' \leftarrow \sum_{(\_,sim,\_)\in\mathcal{W}_{d'}} sim$
  return $\mathcal{W}_{d'}$

windows and the MPSCD is maximized for the new document d'. The resulting MPSCDs in $T(d')$ depend on the initial position of SCD windows, since we consider in the iterative optimization only window adjustments of size one and do not change the number of initially defined SCD windows (M). Thus, the final MPSCDs represent a (local) optimum for the M SCD windows. Finally, Alg. 4 terminates, since the number of possible window adjustments in the algorithm is finite.

Augmenting Enrichment: Document Clusters

If an agent is interested in extending its corpus $\mathcal{D}$ with a document d' containing similar content to documents in $\mathcal{D}$, we can adjust the process for generating SCD for d' ignoring SCDs associated with documents in $\mathcal{D}$ containing content unrelated to the content of d'. The first step in estimating most probably suited SCDs for a corpus-extending document d' consists of identifying those documents in $\mathcal{D}$ having a high topic similarity with d'. We form a cluster $\mathcal{D}_{d'}$ of d'-related documents such that all documents $d \in \mathcal{D}_{d'}$ have a Hellinger distance H of their document-topic probability distributions $\theta_d, \theta_{d'}$ being

smaller than a threshold τ, i.e.,

$$\mathcal{D}_{d'} = \{d \in \mathcal{D} \mid H(\theta_{d'}, \theta_d) < \tau\}. \tag{4.11}$$

The threshold τ decides on the minimum required similarity between two documents d' and d, such that document $d \in \mathcal{D}_{d'}$. The best value for τ depends on the performance of external applications working with the located SCDs in $T(d')$. The document-topic probability distribution of documents in $\mathcal{D}$ changes with respect to α, β, and K in the LDA. The smaller τ is, the higher the topic similarity between the documents in $\mathcal{D}_{d'}$ is. Expression (4.11) requires a document-topic probability distribution $\theta_{d'}$ for the corpus-extending document d' as well as the documents in $\mathcal{D}$. The following three basic strategies exist to arrive at the document-topic probability distributions,

(i) extend corpus $\mathcal{D}$ with new document d' and generate a new topic model from $\mathcal{D} \cup \{d'\}$, which contains document-topic probability distributions θ for each document in $\mathcal{D} \cup \{d'\}$,

(ii) generate a topic model from $\mathcal{D}$, which contains document-topic probability distributions θ for each document in $\mathcal{D}$, and approximate the document-topic probability probability distribution $\theta_{d'}$ using *folding in* Gibbs sampling (Geman and Geman (1984)), or

(iii) generate a topic model from $\mathcal{D}$, which contains document-topic probability distributions θ for each document in $\mathcal{D}$, and approximate the document-topic probability distribution $\theta_{d'}$ using *folding in* Gibbs sampling as in (ii), while also updating the overall topic model, known as the online variational Bayes algorithm for LDA (Hoffman *et al.* (2010)).

In the second part of this dissertation we analyze the three strategies in detail and present advantages and disadvantages for each of the them. However, whichever strategy we use for learning document-topic probability distributions, afterwards we can use the Hellinger distance between document-topic probability distributions of d' and all documents in $\mathcal{D}$ to assemble $\mathcal{D}_{d'}$.

Now, we can perform Alg. 1 with $\mathcal{D}_{d'}$ as input. This leads to a new SCD-word probability distribution $\delta(\mathcal{D}_d)$ which we use as input for Alg. 2 calculating a new set of SCDs for document d'.

Next, we evaluate the performance using document clusters in the second part of the next section.

4.6 Case Study

After introducing the document classification approach to identify the category of a new document based on documents in a corpus, we present a case study illustrating the potential of document classification and both optimization techniques. We use the sequence

of observable MPSCD similarity values as evidence in HMMs and compare the technique against a multi-dimensional classification approach. The multi-dimensional classification approach analyze five different MPSCD similarity properties like the maximum and the minimum similarity value of MPSCDs associated with a new document.

Generally, we can generate corpora by collecting documents we might be interested in. However, the free Wikipedia encyclopedia already contains so called *lists* representing a collection of documents for specific subjects.

So, this case study is based on three corpora, each containing documents from lists of Wikipedia and all corpora contain approximately the same number of words. We have selected exactly those three different corpora, because the topics of all three corpora are different. Generally, our approach works for each corpus. The three corpora are:

- `EU-cities` containing 36 documents about the largest cities in Europe[1].
- `US-presidents` containing 45 documents about presidents of the U.S. between 1789 and 2017[2], and
- `US-universities` containing a subset of 50 documents about the state and territorial universities in the U.S. [3].

4.6.1 Preprocessing

We work with an implementation of Alg. 1, Alg. 2 and Alg. 4 and use their results as input parameters in an implementation of Alg. 3. All documents are preprocessed by: (i) lowercasing all characters, (ii) stemming the words, (iii) tokenizing the result, (iv) eliminating tokens from a stop-word list of 337 words, and (v) performing LDA on the preprocessed documents to generate a topic model from all documents in the corresponding corpus. Generally, SCDs are manually associated with documents. However, the techniques presented in this chapter are independent from the SCD itself and only depends on the corresponding similarity values. Thus, we use Stanford OpenIE (Angeli *et al.* (2015)) to automatically extract (subject,predicate,object)-triples from the documents acting as SCDs for all documents within the corpus.

For Alg. 2 and Alg. 4 only stop-words from the documents are removed. Afterwards, we generate for all four document category and each of the three corpora a training set and test set. Both data tests are generated in the following way:

Similar Documents (sim): 90% of the corresponding Wikipedia documents act as training data, i.e., 90% of the documents represent documents in the corpus. The re-

[1] `https://en.wikipedia.org/wiki/List_of_European_cities_by_population_within_city_limits`
[2] `https://en.wikipedia.org/wiki/List_of_presidents_of_the_United_States`
[3] `https://en.wikipedia.org/wiki/List_of_state_and_territorial_universities_in_the_United_States`

maining 10% of documents from the same corpus are used as a held-out set acting as test data.

Extending documents (*ext*): For each corpus, we choose the training set by selecting the corresponding Wikipedia documents and extend them with text from documents of the other two corpora, s.t. the content of an extending document is unrelated to content in the current corpus. For corpus `EU`, documents from `EU-presidents` and `US-universities` are selected. Documents for the test set are generated in the same fashion.

Revision documents (*rev*): The *view history* function[4] in Wikipedia generates a revised version of a document. The latest version of a document contains modified and additional sentences, since the latest document represents a revision of an older version of the same document. The same steps are performed for creating document in the test set.

Unrelated documents (*unrel*): We randomly choose the training set by selecting 90% of documents from one corpus and randomly selecting documents from the two other corpora to generate a test set. Since the documents in the test set are from different corpora, the content is unrelated to documents in the test set.

For each document category, we take the corresponding training set and learn the transition and emission function for the category-specific HMMs using the Baum-Welch algorithm. This process results in four HMMs each of them representing the configuration for one category of document for each corpus. We use Alg. 3 to perform the document classification of documents in the test set. Precision and Recall represent the performance of Alg. 3, where TP refer to the number of documents whose document category has been correctly estimated, FP refer to the number of documents whose document category has been falsely estimated, and FN refer to the number of documents classified to a category of a document that was not found.

4.6.2 Document Classification

We conduct experiments to test the performance of the document classification approach based on the introduced technique to generate corpora and the evaluation approach. Table 4.1 presents the performance of Alg. 3 using both measures, precision and recall, for all three corpora. Generally, document classification performs best on unrelated and extending documents for the three corpora. The transition probability and emission probability are similar for categories *sim* and *rev* for documents in the corresponding HMMs. The algorithm has never classified revisions and similar documents as unrelated

[4] `https://en.wikipedia.org/wiki/Help:Page_history`

Table 4.1: Document classification performance considering the four document categories *sim*, *unrel*, *ext*, *rev*, and the three corpora *city*, *university*, *president*.

Corpus	Method	Document category			
		sim	*unrel*	*ext*	*rev*
City	Precision	0.72	1.0	0.93	0.70
	Recall	0.65	1.0	0.86	0.41
	F1-Score	0.68	1.0	0.89	0.52
University	Precision	0.77	1.0	0.91	0.72
	Recall	0.71	0.96	0.84	0.58
	F1-Score	0.72	0.98	0.87	0.64
President	Precision	0.70	1.0	0.86	0.68
	Recall	0.71	1.0	0.92	0.51
	F1-Score	0.70	1.0	0.89	0.58

documents or document extensions, respectively, but sometimes the algorithm has classified documents from category *sim* as documents from category *rev*, and vice versa. We assume an agent interested in similar documents might accept document revisions because they are similar to the documents in a corpus. If an agent is only interested in similar documents or revisions, it must analyze documents of both categories in detail, e.g., by using positions of interest. Next, we look at the performance of the optimization techniques.

4.6.3 Window Adaptation

Estimating MPSCDs for new documents requires a predefined number (M) of MPSCDs an agent might be interested in as well as the size of the corresponding MPSCD windows. In Section 3.2, we have described Alg. 2 dividing document d' into M equally sized windows (line 4). Afterwards, Alg. 2 estimates for each of the M windows the MPSCD using SCD-word probability distribution $\delta(\mathcal{D})$. Obviously, the initial segmentation of document d' ignores the content within documents. Thus, we have introduced Alg. 4 adapting the window's size to increase the quality of SCDs causing an optimization of the document classification performance. Left plot in Fig. 4.4 presents the performance of Alg. 4 by comparing the document classification performance based on Alg. 2 and Alg. 4 for documents in the city corpus. The overall classification performance of Alg. 4 is better than the performance of Alg. 2, and the HMM generated from output of Alg. 4 leads to remarkable increase in detecting documents of category *sim* and *rev*. The right plot in Fig. 4.4 presents the average MPSCD similarity value increase for a document after optimizing the window size using Alg. 4. Adapting one window will automatically adapt the neighboring windows and might lead to a decreasing MPSCD similarity value for the neighboring windows. Generally, Alg. 4 adapts the windows in a way that the

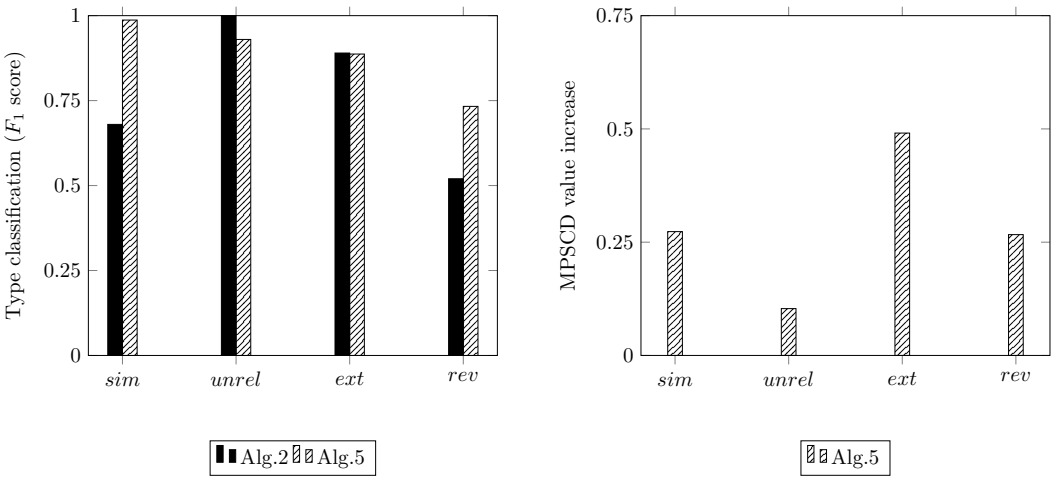


Figure 4.4: The Left plot presents the classification performance using Alg. 2 and Alg. 4. The right plot presents the average MPSCD similarity value increase for each document after optimizing the window size after applying Alg. 4.

sum of all MPSCD similarity values increases and the document category classification performance increases, too.

We use the FDR to evaluate the performance because we are interested in a high number of true positives and small number of false positives. The FDR (see Expression (2.8)) slightly increase using Alg. 4, since the algorithm focus on the optimization of overall MPSCD similarity value increase instead of optimizing the FDR.

4.6.4 MPSCDs for new Documents

Let us assume that an agent is interested in enriching its initial corpus only with similar documents. Then, the agent can use MPSCDs similarity values to identify the category of a new document based on the documents in its corpus. Classifying new documents based on the MPSCDs similarity values might be important for an agent, since the agent has obtained corpus-specific content description for a corpus-extending document based on the most probably suited SCDs. But in general a new document is only eminently related to a subset of documents within a large corpus. Therefore, we are interested in analyzing the quality of subjective content descriptions for a new document d' using only those subset of documents $\mathcal{D}_{d'}$ from corpus $\mathcal{D}$ containing documents having a similar document-topic probability distribution to d'. Again, we use the FDR analyzing the performance of calculated MPSCDs.

In Fig. 4.5, we present the FDR of the MPSCDs corresponding to the MPSCD similarity values used to classify documents. Left plot shows FDR of estimated MPSCDs for documents of category *sim* using the Hellinger distance to estimate the documents in

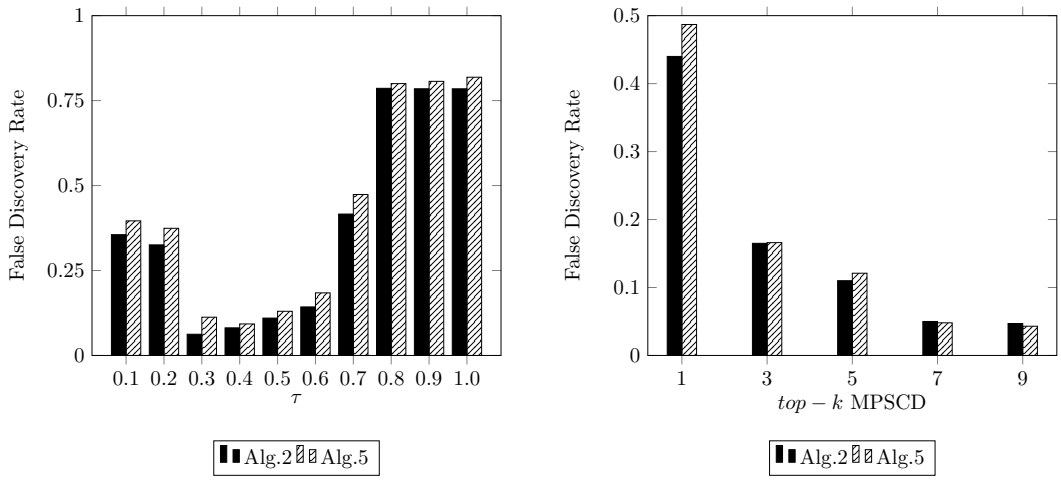


Figure 4.5: Left: False Discovery Rate of MPSCDs for new documents of category *sim* contemplating only documents from the corpus having a Hellinger distance of τ. Right: FDR of MPSCDs for similar new documents contemplating the top-k MPSCDs.

$\mathcal{D}_{d'}$, having a distance less than τ. The FDR is best contemplating only documents having a Hellinger distance between 0.3 and 0.6. Obviously, ignoring a subset of documents from $\mathcal{D}$ changes the SCD-word probability distribution $\delta(\mathcal{D})$. However, contemplating only documents having a low distance ignore the fact that we are approximating the document-topic probability distribution for new documents and documents containing relevant SCDs might not in $\mathcal{D}_{d'}$, having a higher distance than the specified value for τ. The right plot shows the FDR of MPSCDs for documents of category *sim*, too. The average FDR with values for τ (0.3 to 0.5) represent the impact of different top-k MPSCD settings.

4.6.5 Comparison of HMM-based and Multi-dimensional Decision Making

We compare the performance of document classification based on the category-specific HMMs with the performance of a classical multi-dimensional decision making approach analyzing the characteristic of MPSCDs similarity values. In the multi-dimensional decision making approach we consider the following five indicators to decide on extending a corpus with a new document or not:

(i) maximum similarity of all MPSCDs (max. sim.),

(ii) minimum similarity of all MPSCDs (min. sim.),

(iii) maximum difference between highest and lowest similarity value ($\Delta_{max,min}$),

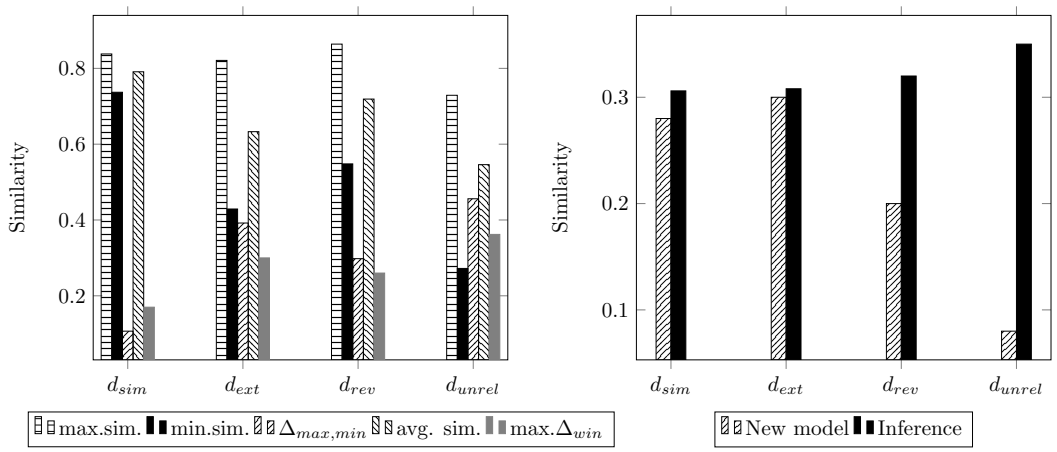


Figure 4.6: Representation of the four document types d_{sim}, d_{ext}, d_{rev}, and d_{unrel} using the introduced MPSCD similarity values (left) and topic similarity (right).

(iv) average MPSCD similarity (avg. sim.), and

(v) maximum change between neighboring MPSCD-windows (max. Δ_{win}).

On the left, Fig. 4.6 presents the indicators for the MPSCDs for each type of document. On the right, Fig. 4.6 presents two variants of comparing a new document with documents in a given corpus using a topic model generated by LDA. The first variant learns a new topic model for the corpus including the new document, yielding a document-topic probability distribution for the new document. The second variant infers the document-topic probability distribution of the new document using an available topic model of the initial corpus. Similarity is given by the Hellinger distance between two document-topic probability distributions subtracted from the value 1.

Inferring the document-topic probability distribution is significantly faster than calculating a new topic model but makes it impossible to identify the type of a new document since all similarity values are similar (see right plot of Fig. 4.6). Generating a new topic model allows for distinguishing an unrelated document and a revision from all types but makes it difficult to distinguish a similar document from an extension because both documents share nearly the same document-topic probability distribution leading to almost identical similarities. Estimating the MPSCDs for a new document and analyzing the indicators enables classification of a new document in the context of a given corpus and is prone to outliers.

The values of indicators are corpus-specific and Table 4.2 presents the five indicators for both corpora. We specify: (i) a high similarity (+) for values between 0.7 and 1, (ii) an average degree of similarity (∘) for values between 0.3 and 0.7, and (iii) and a low

Table 4.2: Document Type Indicator Comparision

	city corpus				president corpus			
	d_{sim}	d_{ext}	d_{rev}	d_{unrel}	d_{sim}	d_{ext}	d_{rev}	d_{unrel}
Max Sim.	+	+	+	∘	+	+	+	∘
Min Sim.	+	∘	∘	−	∘	∘	∘	−
$\Delta_{max,min}$	−	∘	−	∘	−	∘	−	∘
Avg. Sim.	+	∘	+	∘	+	+	+	∘
Max.Δ_{win}	−	∘	−	∘	−	∘	−	∘

similarity (−) for values below 0.3.

For both corpora new documents of type d_{sim}, d_{ext} and d_{rev} share high values for the maximum similarity. Similar documents (d_{sim}) have a noticeable higher minimum similarity than all other types of documents. Unrelated documents (d_{unrel}) have a smaller maximum similarity value compared all other types of documents and the minimum similarity is small. The maximum similarity changes between neighboring windows ($max.\Delta_{win}$) of document extensions (d_{ext}) is similar to unrelated documents while the maximum similarity change between neighboring windows of revisions (d_{rev}) is comparable to similar documents.

In this case study, we have shown the performance of the introduced HMM-based decision making approach classifying a new document given a corpus. Generally, document classification performs best on unrelated and extending documents and it is easy to distinguish similar documents from the revision of a document. The performance on unrelated and extending documents is slightly better than on similar documents and revisions. As shown, in Table 4.2 it is possible to use a multi-dimensional decision making approach to classify new documents with respect to a given corpus simply analyzing five indicators representing specific MPSCD similarity values from the new document. Thus, a multi-dimensional decision making approach has low cost. However, the performance of the HMM-based decision making approach is much better.

Chapter 5

Corpus-Driven Document Enrichment using SCDs

The last chapter has introduced a corpus-controlled decision making approach for new documents based on an estimated set of MPSCDs for new documents given an SCD-word probability distribution matrix resulting from the documents in a corpus. The decision making approach classifies a new document based on the estimated MPSCDs for this new document. A major advantage of the decision making approach is the resulting set of SCDs for new documents extending the initial corpus because an agent can directly use the associated SCDs for additional task. However, the initially associated set of MPSCDs for corpus-extending documents might be insufficient and associating additional SCDs with new documents associated with documents in the corpus might add an additional value to the task of an agent.

Generally, different approaches are available to enrich a document-specific set of SCDs. A well-known approach for associating additional data to a document is Named-Entity Linking (NEL). Shen *et al.* (2015) describe NEL as the task of linking entities available in an external source, e.g., a KB, to entities mentioned in the text of a document. Having identified entities in the text of a document, NEL try to identify for an entity in a document the same entity in a (specific) external source s.t. new entities and relations from the externalsource might be used to associate new data to the document. We argue that using external sources to enrich SCD sets with additional data ignores the corpus-specific context and available SCDs associated with other documents in the corpus as well adds additional data related to NEs mentioned in the text of a document.

In this chapter, we present an approach enriching sparsely and weakly annotated documents with additional SCDs associated with related documents within the same corpus. We assume that associating a document with SCDs from other documents of the same corpus might add a value for the tasks of an agent, since a corpus represents a specific context resulting in context-specific SCD enrichment. We introduce two similarities comparing documents from the same corpus with each other to decide if two documents are somehow related. The first similarity, denoted as *D-similarity*, estimates the similarity between two documents using the textual content of both documents. The second similarity, denoted as *T-similarity*, compares two documents at the SCD-level. To enrich

sparsely and weakly annotated documents with SCDs associated with documents from the same corpus we introduce an unsupervised EM-like algorithm using a combination of both, the D- and T-similarity. First, the algorithm identifies for a document the set of related documents. Second, the algorithm assigns each SCD a document-specific Expected Relevance Value (ERV) representing the relevance of an SCD for the document.

5.1 Similarities

In this section, we introduce the D-similarity and T-similarity. The D-similarity estimates the similarity between two documents based on both per-document topic probability distributions. The T-similarity between two documents results from the matching SCDs associated with both documents.

5.1.1 D-Similarity

As described in Section 2.2, ML-based text-mining approaches denoted as topic models try to identify hidden semantic structures within the text of documents relating a set of observed multivariate variables to a set of hidden classes. A document can be represented in a corpus using a corpus-specific document-topic probability distribution θ. The D-similarity is based on the idea of topic models and compares the similarity of two documents by their document-topic probability distributions.

We use LDA, introduced in Section 2.3, to calculate a corpus-specific topic model and calculate for each document $d \in \mathcal{D}$ the corresponding document-topic probability distribution θ_d. The document-topic probability distribution θ_d of document d is represented as a *topic vector*.

We define the D-Similarity between document d_e and d_k, both documents from the same corpus by:

$$Sim_D(d_e, d_k) = 1 - H(\theta_{d_e}, \theta_{d_k}), \tag{5.1}$$

where $H(\theta_{d_e}, \theta_{d_k})$ estimates the Hellinger distance between the document-topic probability distribution of d_e and document-topic probability distribution of d_k. The interval of D-similarity $Sim_D(d_e, d_k)$ follows directly from the definition of the Hellinger distance s.t. $Sim_D(d_e, d_k) \in [0, 1]$. The higher the D-similarity between two documents d_e and d_k the more similar the document-topic probability distributions of both documents. If the D-similarity $Sim_D(d_e, d_k)$ of d_e and d_k is high it follows that both documents contain similar content, since each topic is represented by a specific word probability distribution over the vocabulary $\mathcal{V}_\mathcal{D}$. We assume that associating SCDs of document d_k to document d_e might add a value for the task of an agent.

Comparing two documents with each other using the D-similarity requires both documents to have the same underlying topic model. Generally, we can estimate a topic

model from all documents in a corpus. But, how to compare the document-topic probability distributions of two documents with each other using the D-similarity if document $d \in \mathcal{D}$ and a new document $d' \notin \mathcal{D}$?

The simplest approach is to add d' to $\mathcal{D}$ and then estimate a new topic model from all documents in the extended corpus. Afterwards, it is possible to compare documents by their document-topic probability distribution. However, this approach is expensive because generating a new topic model is time-consuming.

Thus, in Chapter 9, we describe different techniques to compare $d \in \mathcal{D}$ with $d' \notin \mathcal{D}$ using the D-similarity without estimating a new topic model all the time a new document occurs.

5.1.2 T-Similarity

The T-similarity works at SCD-level and identifies for a document $d_e \in \mathcal{D}$ a set of d_e-related documents in corpus $\mathcal{D}$ based on an ERV of associated SCDs. For each document $d \in \mathcal{D}$ exists an SCD set $T(d)$ containing all SCDs associated with document d. As defined in Definition 2.1.6, an SCD might be represented as a triple containing a subject (s), predicate (p), and an object (o). We are interested in comparing two documents with each other using their associated SCDs. Thus, we introduce the T-similarity representing the similarity between two documents from the same corpus based on their associated SCDs following the idea that semantically related documents share at least a subset of SCDs or at least parts of SCDs like subject, object, or the relation between them.

First, we define a similarity function $s(t_i, t_j)$ calculating a similarity score between two SCDs t_i and t_j. We use the similarity function $s(t_i, t_j)$ to compare the i-th SCD in $T(d_e)$ with the j-th SCD in $T(d_k)$ using the entities and relations to estimate a similarity score in $[0, 1]$. The more similar two SCDs $t_i \in T(d_e)$ and $t_j \in T(d_k)$ the higher the returned value of similarity function $s(t_i, t_j) \in [0, 1]$.

We define similarity function $s(t_i, t_j)$ by:

$$s(t_i, t_j) = \begin{cases} 0 & \text{if}(s_i \neq s_j \wedge p_i \neq p_j \wedge o_i \neq o_j) \\ \frac{1}{3} & \text{if } (s_i = s_j \wedge p_i \neq p_j \wedge o_i \neq o_j) \vee \\ & (s_i \neq s_j \wedge p_i = p_j \wedge o_i \neq o_j) \vee \\ & (s_i \neq s_j \wedge p_i \neq p_j \wedge o_i = o_j), \\ \frac{2}{3} & \text{if } (s_i = s_j \wedge p_i = p_j \wedge o_i \neq o_j) \vee \\ & (s_i \neq s_j \wedge p_i = p_j \wedge o_i = o_j) \vee \\ & (s_i = s_j \wedge p_i \neq p_j \wedge o_i = o_j), \\ 1 & \text{if } (s_i = s_j \wedge p_i = p_j \wedge o_i = o_j) \end{cases} \tag{5.2}$$

The similarity function does not weight any part of an SCD and increases the similarity value for each matching part between two SCDs. Generally, depending on the specific

form of SCDs, the similarity function needs to be adopted.

Calculating the T-similarity between SCD set $T(d_e)$ and SCD set $T(d_k)$ requires an SCD-wise comparison of each SCD $t_i \in T(d_e)$ with all SCDs in $T(d_k)$ using the similarity function $s(t_i, t_j)$ for all $t_i \in T(d_e)$ and all $t_j \in T(d_k)$.

$$\mathcal{M}_T = \begin{array}{c} \\ t'_1 \\ t'_2 \\ \vdots \\ t'_m \end{array} \begin{array}{c} \begin{array}{ccccc} t_1 & t_2 & t_3 & \cdots & t_n \end{array} \\ \begin{pmatrix} a_{1,1} & a_{1,2} & a_{1,3} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & a_{2,3} & \cdots & a_{2,n} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{m,1} & a_{m,2} & a_{m,3} & \cdots & a_{m,n} \end{pmatrix} \end{array} \tag{5.3}$$

Matrix $\mathcal{M}_T$ (Expression (5.3)) is an $m \times n$ matrix, where m is the number of rows and n is the number of columns. $\mathcal{M}_T$ represents all possible similarity scores between SCDs in $T(d_e)$ and $T(d_k)$ where $m = |T(d_e)|$ and $n = |T(d_k)|$, such that $a_{i,j}$ represents the similarity score for $s(t_i, t_j)$, $t_i \in T(d_e)$, $t_j \in T(d_k)$. Since, $t_1 \in T(d_e)$ does not necessarily be the same SCD as $t_1 \in T(d_k)$ we mark all SCDs associated with d_e with a dash. Hence, we use $\mathcal{M}_T$ to identify the best match for each SCD in $T(d_e)$ and all SCDs in $T(d_k)$, and vice versa.

Next, we define similarity vectors used to estimate the T-similarity between both SCD sets using the values in $\mathcal{M}_T$

Definition 5.1.1 (similarity vector)**.** We define the following two similarity vectors v^c and v^r:

$v^c \in \mathbb{R}^n$ with $v^c_j = max_i\ a_{i,j}$ representing a similarity vector containing for each SCD in $T(d_e)$ the highest possible similarity score with respect to the SCDs in $T(d_k)$ and

$v^r \in \mathbb{R}^m$, with $v^r_i = max_j\ a_{i,j}$ representing the similarity vector containing for each SCD in $T(d_k)$ the highest possible similarity score with respect to the SCDs in $T(d_e)$.

Example 5.1.1 represents a calculation to estimate the similarity vectors v^c and v^r.

Example 5.1.1 (similarity vector)**.** Let us assume we have an SCD set $T(d_e)$ containing SCDs t_1, t_2, and t_3 and an SCD set $T(d_k)$ containing SCDs t'_1, t'_2, and t'_3. Then, matrix $\mathcal{M}_T$ represents values from the similarity function defined in Expression (5.2).

$$\mathcal{M}_T = \begin{array}{c} \\ t'_1 \\ t'_2 \\ t'_3 \end{array} \begin{array}{c} \begin{array}{ccc} t_1 & t_2 & t_3 \end{array} \\ \begin{pmatrix} \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ \frac{2}{3} & \frac{1}{3} & \frac{3}{3} \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \end{pmatrix} \end{array} \tag{5.4}$$

We estimate similarity vectors v^c and v^r using the similarity values in Expression (5.4): Both vector v^c and v^r contain three entries. Vector v^c contains the maximum of each column and v^r contains the maximum of each row.

$$v_1^c = max\{\frac{1}{3}, \frac{2}{3}, \frac{1}{3}\} = \frac{2}{3} \qquad v_1^r = max\{\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\} = \frac{1}{3}$$
$$v_2^c = max\{\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\} = \frac{1}{3} \qquad v_2^r = max\{\frac{2}{3}, \frac{1}{3}, \frac{3}{3}\} = \frac{3}{3}$$
$$v_3^c = max\{\frac{1}{3}, \frac{3}{3}, \frac{1}{3}\} = \frac{3}{3} \qquad v_3^r = max\{\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\} = \frac{1}{3}$$

Definition 5.1.2 (T-similarity)**.** We use both similarity vectors v^c and v^r to calculate the T-similarity which is defined by

$$Sim_T(T(d_e), T(d_k)) = \frac{1}{2} \cdot (\overline{v^c} + \overline{v^r}), \tag{5.5}$$

where $\overline{v^c}$ and $\overline{v^r}$ represent the average value of similarity vectors taking the ratio between high and low similarity scores into account such that two SCD sets $T(d_e)$ and $T(d_k)$ sharing only a small number of high similarity values and a high number of low similarity scores for SCDs resulting in a small T-similarity for $T(d_e)$ and $T(d_k)$, respectively. We normalize $Sim_T(T(d_e), T(d_k))$ to the interval $[0, 1]$.

Example 5.1.2 (T-similarity)**.** We use matrix $\mathcal{M}_T$ from Example 5.1.1 and calculate $\overline{v^c}$ and $\overline{v^r}$ resulting in:

$$\overline{v^c} = \frac{\frac{2}{3} + \frac{1}{3} + \frac{1}{3}}{3} = \frac{4}{9} \qquad \overline{v^r} = \frac{\frac{1}{3} + \frac{3}{3} + \frac{1}{3}}{3} = \frac{5}{9}$$

Now, we can directly calculate the T-similarity between $T(d_e)$ and $T(d_k)$ by:

$$Sim_T(T(d_e), T(d_k)) = \frac{1}{2} \cdot (\overline{v^c} + \overline{v^r}) = \frac{1}{2} \cdot (\frac{4}{9} + \frac{5}{9}) = \frac{1}{2}$$

5.1.3 Expected Relevance Value

We use both, the D- and T-similarity to define the ERV for SCDs representing the relevance of an SCD for a document an agent is interested in associating with additional SCDs. We represent the set of all documents in $\mathcal{D}$ being related to document d_e by $\mathcal{D}_{d_e}$ and use $T(\mathcal{D}_{d_e})$ referring to the set of all SCDs associated with documents in $\mathcal{D}_{d_e}$. For each document $d \in \mathcal{D}_{d_e}$ we are interested in the ERV of its associated SCDs representing the relevance of an SCD for document d_e. We define the ERV of SCDs to estimate only those SCDs in $T(\mathcal{D}_{d_e})$ having a high ERV.

Definition 5.1.3 (Expected Relevance Value)**.** The Expected Relevance Value of SCD $t \in T(\mathcal{D}_{d_e})$ is defined by:

$$ERV_t^{d_e} = \overline{Sim_{D_t^{d_e}}} \cdot \overline{Sim_{T_t^{d_e}}} \cdot f(t), \tag{5.6}$$

where $\overline{Sim_{D_t^{d_e}}}$ represents the average D-similarity of all documents in $\mathcal{D}_{d_e}$ being associated with SCD t, $\overline{Sim_{T_t^{d_e}}}$ represents the average T-similarity of all SCD sets associated with documents in $\mathcal{D}_{d_e}$ containing SCD t, and $f(t)$ represents the number of occurrences of an SCD t associated with documents in $\mathcal{D}_{d_e}$, denoted by $T(\mathcal{D}_{d_e})$.

Since there are three terms in Expression (5.6) the following three settings yielding to a high ERV of an SCD t.

(i) D- and T-similarity between document d_e and documents in $\mathcal{D}_{d_e}$ is high which means the content of each $d \in \mathcal{D}_{d_e}$ might be similar to d_e, because the document-topic probability distribution as well as the SCDs associated with both documents is similar.

(ii) Many documents in $\mathcal{D}_{d_e}$ being associated with SCD t.

(iii) The frequency of an SCD t, since enriching the SCD set $T(d_e)$ of document d_e with SCD t associated with many other documents may add value, because the SCD might be a generic SCD, or a very specific SCD for the d_e-related documents.

Definition 5.1.3 depends on the SCDs we are interested in and the specific task of an agent using the SCDs, s.t. the ERV might be slightly adopted for the individual task of an agent.

In the next section, we use both similarities and the ERV of SCDs to enrich documents in a corpus that are only sparsely and weakly annotated.

5.2 Iterative SCD Enrichment Algorithm

We present an unsupervised and iterative SCD enrichment algorithm estimating additional SCDs for a document based on SCDs associated with related documents in the same corpus. The algorithm is based on the famous EM-algorithm of Dempster *et al.* (1977). Dempster *et al.* (1977) have introduced the EM-algorithm for estimating the maximum likelihood of parameters handling unobserved variables alternating between an expectation and maximization step, where the expectation step creates a function for the expectation of log-likelihood using the present values for the parameters and the maximization step calculates the parameters maximizing the expected log-likelihood in the expectation step.

Algorithm 5 introduces an EM-like SCD enrichment algorithm enriching the SCD set of a document with SCDs of related documents of the same corpus by the following three steps:

(i) Identifying for each document $d \in \mathcal{D}$ a set of related documents ($\mathcal{D}_d$) using both, D- and T-similarity, where the D-similarity estimates the similarity between two documents using their document-topic probability distributions and the T-similarity works at the SCD-level comparing SCDs.

(ii) Iteratively enriching the SCD set of documents in corpus $\mathcal{D}$ with additional SCDs associated with related documents.

(iii) Annotating new and unseen documents using the SCDs associated with related documents, and vice versa.

The input parameters of Alg. 5 are: (i) document d_e, (ii) the corresponding SCD set $T(d_e)$, (iii) corpus $\mathcal{D} \setminus \{d_e\}$, and (iv) D-similarity selection threshold τ.

The output of Alg. 5 is an enriched SCD set for document d_e represented as $T'(d_e)$. Using the E-set and M-step, Algorithm 5 adds only SCDs with high ERV to the SCD set $T'(d_e)$ and ignores SCDs with low ERV.

Expectation step. For document d_e the expectation step in Alg. 5 identifies all related documents ($\mathcal{D}_{d_e}$) based on the average T-similarity ($\overline{Sim_{T^{d_e}}}$) of documents in $\mathcal{D}$ and the D-similarity between new document d_e and each documents in $\mathcal{D}$. Additionally, Alg. 5 estimates for each SCD t, associated with the d_e-related documents, the corresponding ERV (erv_t) using Expression (5.6).

Maximization step. In the maximization step, Alg. 5 calculates a new average T-similarity $\overline{Sim_{T^{d_e}}}$ (line 17) optimizing the ERVs for SCDs in the next expectation step after expanding the SCD set associated with d_e with SCDs associated with related documents. The average T-similarity of all SCDs is part of the termination condition in line 5.

Algorithm 5 enriches SCD sets in a corpus-controlled fashion and terminates for each $\epsilon \in \mathbb{R}^+$ and a finite set of documents in $\mathcal{D}$.

Let ϵ be greater than zero and $\mathcal{D}$ be a finite set. Then, Alg. 5 terminates if the condition in line 6 is not fulfilled anymore. In the M-step, Alg. 5 updates the average T-similarity $\overline{Sim_{T^{d_e}}}$. This approach prunes the solution space for the next iteration because the new average T-similarity leads to a more stringent condition in line 10 compared to the previous iteration. In each iteration $\mathcal{D}_{d_e}$ represents the set of documents such that $T(\mathcal{D}_{d_e})$ contains only SCDs having a T-similarity greater than the average T-similarity. In each iteration we optimize the SCDs in $T(\mathcal{D}_{d_e})$ calculating for each of the SCDs a new ERV. Hence, the algorithm extends $T'(d_e)$ only with SCDs best describing the content of d_e.

Algorithm 5 Iterative SCD Enrichment

1: **function** IterativeSCDEnrichment(d_e, $T(d_e)$, $\mathcal{D}$, τ)
2: **Input**: document d_e, SCD set $T(d_e)$, corpus $\mathcal{D}$, threshold τ
3: **Output**: Enriched SCD set $T(d_e)$
4: **Define:** $\epsilon = 0.1$, $\mathcal{D}_{d_e}$, $\mathcal{D}'^{d_e}$, $T(\mathcal{D}_{d_e})$, $T'(d_e)$
5: **Initialize:** $\overline{Sim_{T^{d_e}}} = \epsilon$, $\overline{Sim'_T} = \overline{Sim_{T^{d_e}}} - \epsilon$
6: **while** $|\overline{Sim_{T^{d_e}}} - \overline{Sim'_{T^{d_e}}}| \geq \epsilon$ **and** $\overline{Sim_{T^{d_e}}} > \overline{Sim'_T}$ **do**
7: $\mathcal{D}_{d_e} \leftarrow \emptyset$ ▷ E-Step
8: **for each** $d_k \in \mathcal{D} \setminus \{d_e\}$ **do**
9: **if** $Sim_D(d_e, d_k) > \tau$ **and** $Sim_T(T(d_e), T(d_k)) > \overline{Sim_{T^{d_e}}}$ **then**
10: $\mathcal{D}_{d_e} \leftarrow \mathcal{D}_{d_e} \cup \{d_k\}$
11: **for each** $t \in T(\mathcal{D}_{d_e})$ **do**
12: $erv_t^{d_e} \leftarrow ERV_t^{d_e}$
13: **for each** $t \in T(\mathcal{D}_{d_e})$ **do**
14: **if** $erv_t^{d_e} > \overline{erv}^{d_e}$ **then**
15: $T(d_e) \leftarrow T(d_e) \cup \{t\}$
▷ M-Step
16: $\overline{Sim'_{T^{d_e}}} = \overline{Sim_{T^{d_e}}}$
17: $\overline{Sim_{T^{d_e}}} = \frac{\sum_{k=1}^{|\mathcal{D}_{d_e}|} Sim_{T^{d_e}(T(d_e), T(d_k))}}{|\mathcal{D}_{d_e}|}$
18: **return** $T(d_e)$

Complexity. The complexity of Alg. 5 depends only on the number of documents in the corpus $\mathcal{D}$ (D) and the number of SCDs in $T(\mathcal{D})$ (m). For each document $d \in \mathcal{D}$ we are interested in enriching its SCD set with SCDs from the SCD set of related documents. Algorithm 5 estimates for each document the set of related documents using the D- and T-similarity and extends $T(d_e)$ only with the most similar SCDs in $T(\mathcal{D}_{d_e})$ instead of enriching $T(d_e)$ with all SCDs associated with the d_e-related documents. This step is important to reduce the number of SCDs in $T(d_e)$ adding *no value* to the task of an agent. The worst case complexity for calculating for each document $d \in \mathcal{D}$ both similarities is in $\mathcal{O}(m^2 D)$ *(line 9)*. Enriching SCD set $T(d)$ for each document d in $\mathcal{D}$ *(line 11 to 12)* has worst case complexity $\mathcal{O}(m^2 D^2)$.

Algorithm 5 calculates for $D - 1$ documents the D-similarity $Sim_D(d_e, d_k)$ and T-similarity $Sim_T(T(d_e), T(d_k))$, where the D-similarity has a complexity $\mathcal{O}(1)$ given the document-topic probability distributions in $\mathcal{D}$. The constant complexity is given by the number of K multiplications of discrete document-topic probability distributions in the D-similarity. T-similarity has a worst case complexity of $\mathcal{O}(m^2)$ because it is theoretically possible that $T(d_e)$ and $T(d_k)$ might contain all SCDs in $T(\mathcal{D})$.

In practice, the document-specific SCD set contains only few SCDs having a relation to the content of the related document. Hence, each $T(d_i)$ is small and $|T(d_i)| << m$. Additionally, for each t in $T(\mathcal{D}_{d_e})$ Alg. 5 calculates the ERV in constant time $\mathcal{O}(1)$ *(line 12)*. Afterwards, Alg. 5 filters all SCDs having an ERV less than the average ERV $\overline{ERV}^{d_e}$

(line 14 to 15). For both steps, the algorithm iterates over $T(\mathcal{D}_{d_e})$ having a theoretically size of m SCDs. This leads to complexity $\mathcal{O}(m)$.

Hence, worst case complexity for calculating the SCDs for each document in a corpus is given by $\mathcal{O}(m^2 D)$. In total, Alg. 5 has a worst case complexity for estimating the SCDs of one document of $\mathcal{O}(D^2 m^2)$. Using Alg. 5 for each document in $\mathcal{D}$ to update the document-specific SCDs leads to the complexity $\mathcal{O}(D^3 m^2)$.

However, in practice the number of documents $d_k \in \mathcal{D}$ (D'), being similar to document d_e, is small ($D' \ll D$) and the rank of the similarity matrix $\mathcal{M}_T$ (m') is small, too ($m' \ll m$). Additionally, the number of iterations for each document is only a fraction of D (see Section 5.3).

5.3 Case Study

In this section, we present empirical results of the introduced corpus-driven document enrichment approach enriching sparsely and weakly annotated documents in a corpus with additional SCDs associated with related documents of the same corpus.

Generally, we might generate corpora by collecting documents we are interested in. However, the free Wikipedia encyclopedia already contains so called *lists* representing a collection of documents for specific subjects. So, in this case study we use the following three corpora, each containing documents from lists of Wikipedia:

- Dataset 1 (`BMW`): Documents related to cars from BMW.
- Dataset 2 (`Mercedes`): Documents related to cars from Mercedes.
- Dataset 3 (`US`): Documents from US universities and institutions.

More details about the documents we use in each dataset are available in Appendix B.

Corpus $\mathcal{D}$ contains for each dataset a set of documents from Wikipedia and each document represents one article from Wikipedia. Additionally, we use DBpedia (Lehmann *et al.* (2014)) to add data to the SCD set associated to each document in $\mathcal{D}$. The data in the so called KB from DBpedia is extracted in a crowd-sourced community effort and mostly about structured content like infoboxes and tables within the Wikpedia articles. This means if document d_e contains text from the article *car* then it follows that $T(d_e)$ contains the corresponding data of entity *car* from the DBpedia KB.

The goal of the corpus-driven SCD enrichment algorithm is identifying additional SCDs for a document $d \in \mathcal{D}$ that might add a value for the task of an agent. We assume that the data available in DBpedia is a good starting point to describe the content of Wikipedia articles, because DBpedia contains data extractable from structured content like infoboxes and tables within the Wikpedia articles. Obviously, we might use any information extraction system as a starting point to obtain content descriptions directly extractable

from the Wikipedia articles. However, the data from DBpedia have a high quality and we are able to use them directly from the DBpedia KB instead integrating third party software. Next, we give a brief overview of the preprocessing steps for Wikipedia articles including details about the hyperparameters for the D- and T-similarity.

5.3.1 Data Preprocessing

We implement Alg. 5 and use the `MALLET` library, introduced by McCallum (2002), for topic modeling with the following parameters: (i) $\alpha = 0.01$, (ii) $\beta = 0.01$, and (iii) 1000 iterations for the model in library `MALLET`. We analyze the performance of Alg. 5 using different settings for the number of topics within a topic model ($k = 5, 10, 20, 30$) to identify the impact of parameter k, representing the number of topics within the model. Generally, the number of topics within the topic model leading to best results is highly dependent on the documents in corpus $\mathcal{D}$.

`MALLET` can be used to preprocess all text documents in corpus $\mathcal{D}$ by: (i) lowercasing all characters, (ii) stemming words, (iii) tokenizing the result, and (iv) eliminating tokens part of a stop-word list which contains 524 words.

All documents in each corpus, represented by each dataset, require additional preprocessing using the following steps:

(i) Analyzing the data in each SCD set and marking SCDs occurring in at least two SCD sets of two different documents such that Alg. 5 can theoretically enrich the SCD set $T(d_e)$ of document d_e with *correct* SCDs meaning that even if we remove some SCDs from $T(d_e)$, Alg. 5 has a chance to identify those removed SCDs within at least one SCD sets of another document in $\mathcal{D}$.

(ii) The second step is about test set generation and contains two sub-steps: (a) Choose one document $d_e \in \mathcal{D}$ and remove 90% of SCDs ($r = 0.9$) in the corresponding SCD set $T(d_e)$ by removing 90% of SCDs which are associated with at least one other document in $\mathcal{D}$ and removing 90% of SCDs which are only available in $T(d_e)$. (b) Generating $|\mathcal{D}|$ different corpora $\mathcal{D}_1$ to $\mathcal{D}_{|\mathcal{D}|}$, each containing $|\mathcal{D}| - 1$ documents and generating for each of the $|\mathcal{D}|$ different corpora a new topic model.

(iii) Inferring the document-topic probability distribution for document d_e using parameters of the corresponding topic model from documents $\mathcal{D} \setminus \{d_e\}$ generated in the second step. Inference bases on the *folding in* Gibbs sampling technique. Topic probability distribution θ_{d_e} of document d_e enables us to compare document d_e with other documents from the same corpus using the D-similarity.

Next, Alg. 5 estimates for each document d_e the SCDs in $T(\mathcal{D}_{d_e})$ associated with d_e-related documents and iteratively enriches each SCD set $T(d_e)$ with SCDs associated with other documents in $\mathcal{D}_{d_e} \setminus \{d_e\}$ having a high ERV. Alg. 5 is performed $|\mathcal{D}|$ times to use each of the generated corpora.

5.3.2 SCD Enrichment

The following four steps are performed to enrich SCD set $T(d_e)$ of document d_e with data from d_e-related documents:

(i) Estimating the document-topic probability distribution θ_{d_e} of document d_e.

(ii) Identifying d_e-related documents based on D- and T-similarity.

(iii) Generating a set $\mathcal{D}_{d_e}$ containing all d_e-related documents.

(iv) Calculating the Expected Relevance Value for each $t \in T(\mathcal{D}_{d_e})$ with respect to d_e.

Afterwards, Alg. 5 optimizes the SCD set $T(d)$ for each $d \in \mathcal{D}$ by enriching the SCD set with new SCDs associated with related documents from the same corpus.

First, we evaluate the performance of Alg. 5 using the following three measures: (i) True Positive Rate (TPR),(ii) Positive Predictive Value (PPV), and(iii) F-measure.

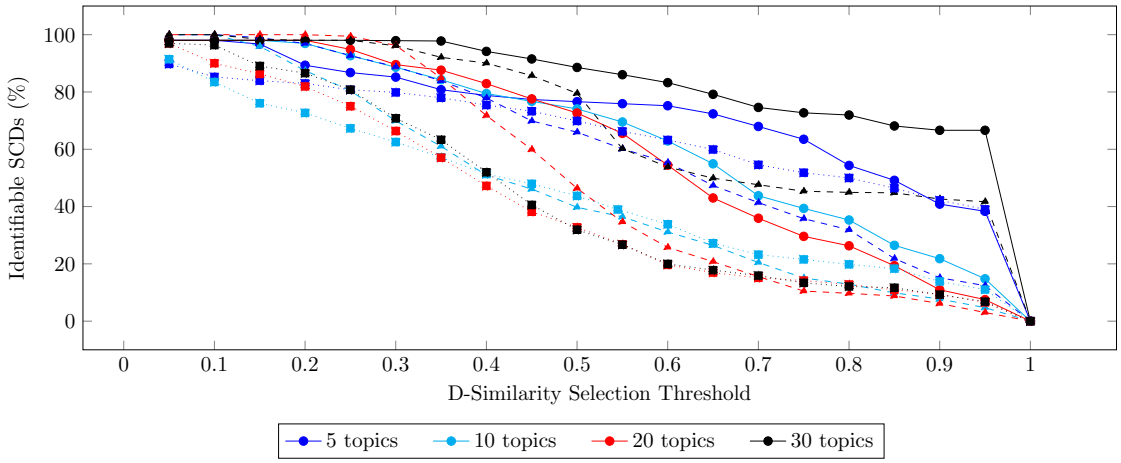
Second, we analyze the required number of iterations in Alg. 5 yielding to the optimal SCDs for each document.

Number of topics. As we have assumed, the performance of Alg. 5 highly depends on the number of topics we use in the topic model for the corpus and the optimal number of topics leading to best results depends on the documents representing the corpus. For datasets `BMW` and `Mercedes`, Alg. 5 has best results using 30 topics. Interestingly, for both datasets the SCD performance using only 5 topics varies only slightly from the results using 30 topics. For dataset `US`, we have best results using $k = 30$ and a D-similarity selection threshold between 0.01 and 0.25. For higher D-similarity selection thresholds, a number of 5 topics results in a better performance.

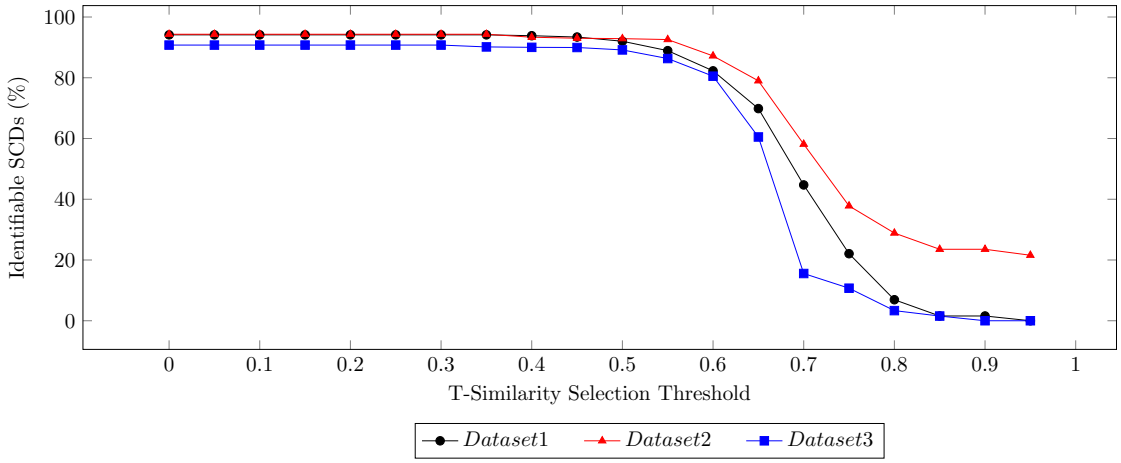
Similarities. Identifying d_e related documents using the T-similarity compares the documents on the SCD level using SCDs in the SCD set. Figure 5.1 presents the influence of both similarities, the D-similarity (Fig. 5.1a) and T-similarity (Fig. 5.1b), within the SCD enrichment process for the `BMW` dataset. Algorithm 5 identifies for the dataset the possible number of SCDs using a T-similarity selection threshold which is at most using a T-similarity between 0.0 and 0.6. Additional plots are available in A.1.

Figure 5.2 presents the number of iterations for all documents in the three datasets having a fixed D-similarity ($Sim_D = 0.25$) and fixed number of topics ($k = 30$), considering only those documents fulfilling the condition in line 5 of Alg. 5 at least once. The median for all three datasets is less or equals five iterations meaning that Alg. 5 quickly reaches a fixpoint and terminates.

Next, we analyze the performance of Alg. 5 for varying D-similarity selection thresholds. Figure 5.3 presents the performance for all three datasets using the following parameters: $k = 30$, $r = 0.90$, $\epsilon = 0.01$, $\overline{Sim_{T^{d_e}}} = 0.1$. The TPR increases for all



(a) Influence of D-Similarity identifying SCDs for documents in `BMW` (lines), `Mercedes` (dashed), and `US` (dotted).



(b) Influence of T-similarity identifying SCDs for documents in all three datasets.

Figure 5.1: Influence of D- and T-similarity in identifying SCDs.

datasets up to a selection threshold of 0.7 while PPV is slightly decreasing at the same time. F-Base is the baseline representing F-measure of random guesses.

The PPV in Fig. 5.3 increases with increasing D-similarity threshold for all three datasets. The TPR seems to be low, but the reason for the low Positive Predictive Value is the data used as ground truth. Algorithm 5 enriches the SCD set of documents with SCDs from related documents which are not part of the documents corresponding

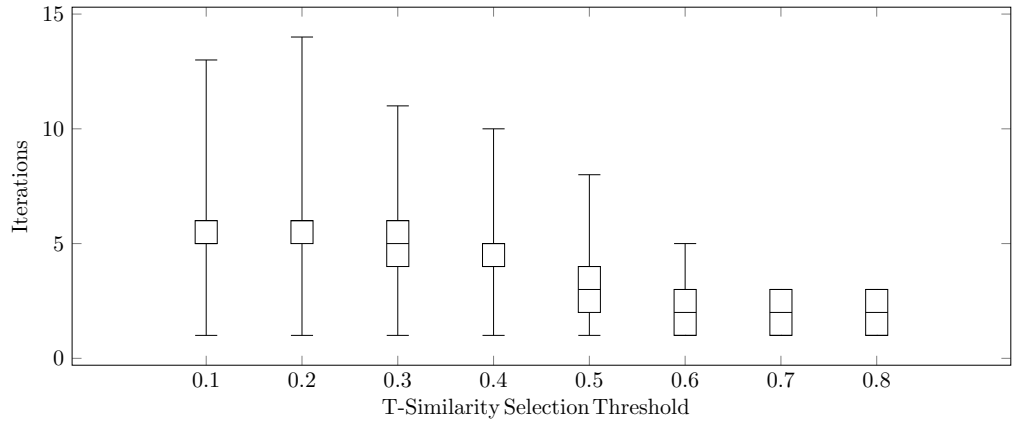


Figure 5.2: Number of necessary iterations for documents in dataset `BMW` until Alg. 5 reaches a fixed point, depending on different input values for the T-similarity.

DBpedia entry. Associating SCDs with a document which are not part of the SCD set we use as ground truth leads to a high number of false negative SCDs within the evaluation. However, we are interested in enriching documents with SCDs adding a value for the task of an agent the SCD set of a document and we are not interested in associating only SCDs to an SCD set of a document resulting in exactly the same SCDs available in the DBpedia KB, since DBpedia ignores the task of an agent and the individual collection of documents. Generally, it is not possible to have only one correct set of subjective content descriptions for a document, even if two different human experts would annotate the same document. One reason is given by the different tasks of different annotators annotating a document and another reason is given by the different background knowledge of both experts. Thus, only a subset of false negative SCDs might add *no* value for the task of an agent working with a set of documents.

5.3.3 Associative Subjective Content Descriptions

Algorithm 5 enriches the SCD set of documents with SCDs associated with related documents. After performing Alg. 5, we have identified SCDs describing the content of the corresponding documents but are not available in DBpedia. We denote those kind of SCDs as associative SCDs. Thus, associative SCDs describe the content of a document, but the SCDs not linked to the corresponding entity in DBpedia's KB acting as ground truth in this case study. Obviously, associative SCDs have a relation to the content of documents and enrich SCD sets with possibly valuable SCDs based on the set of documents within a given corpus. Table 5.1 presents some SCDs identified by Alg. 5 which are not extractable by DBpedia's KB for entities *M41*, *1_Series*, *Z3*, and *M_Roadster*.

However, some of the SCDs are suitably content descriptions for the corresponding documents and represent associative SCDs. In the evaluation of Alg. 5, we mark asso-

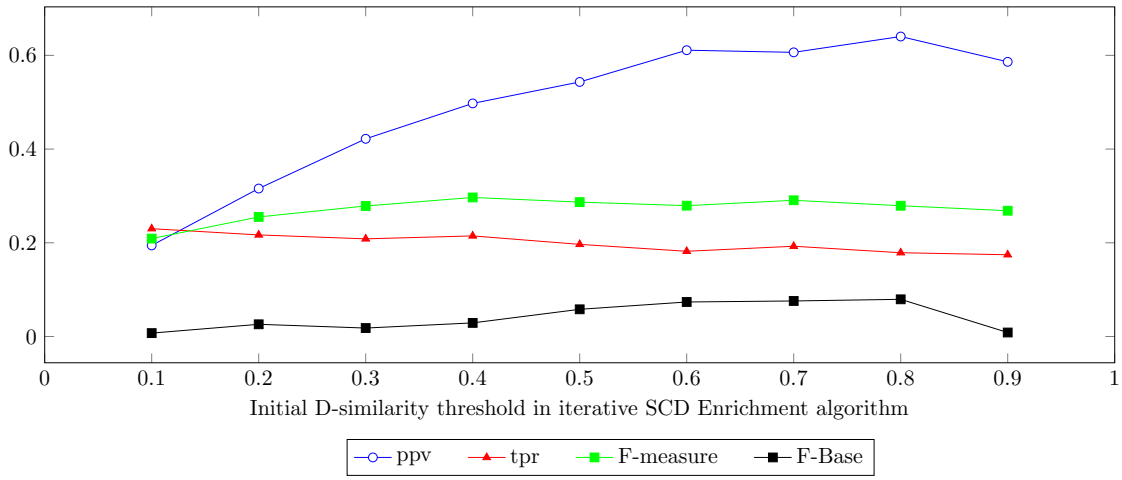


Figure 5.3: Performance depending on D-similarity for $k=30$, $r=0.90$, $\epsilon=0.01$, $\overline{Sim_{T^{de}}} = 0.1$. F-Base represents the F_1-score for randomly guessing possible SCD for the SCD set for the corpus containing documents relating to the car brand *BMW*.

ciative SCDs as false positive SCDs, because they are not in the data acting as ground truth. However, a large number of false positive SCDs represent associative SCDs adding a value for different tasks of an agent, since the SCDs describe the documents' content. Thus, the results would have been better in practice if human experts had rated the added value of each SCD, but generally, we are not interested in generating the same data available in DBpedia, since we involve the goal of an agent within the enrichment process.

doc_17484	**doc_7071**
M41,subject,Category:BMW_vehi.	Z3,hypernym,Car
M41,hypernym,Car	Z3,hypernym,Engine
	Z3,configuration,Diesel_engine
doc_3407	**doc_45969**
1_Series,subject,Cat.:Roadsters	M_Roadster,bodyStyle,Coupe
1_Series,conf.,Diesel_engine	M_Roadster,hypernym,Car
1_Series,hypernym,Engine	M_Roadster,hypernym,Engine
	M_Roadster,subject,cat:1960s_auto.

Table 5.1: Example of identified SCDs in `BMW` using Alg. 5.

BMW Z3 The content of document d_e (*doc_ 7071*) describes the two-seat convertible and coupe *Z3* from car manufacturer BMW. Algorithm 5 enriches the SCD set $T(d_e)$ with SCDs (BMW_Z3, hypernym, Car), (BMW_Z3, hypernym, Engine), and (BMW_Z3, configuration, Diesel_engine). Obviously, the model Z3 is a car, but the authors of the article do not explicitly write that BMW Z3 is a car even though the word *car* occurs 8 times in document d_e. Enriching SCD set $T(d_e)$ with SCD (BMW_Z3, hypernym, Car) is beneficial for people searching for car models manufactured by BMW.

The other two associative SCDs (BMW_Z3, hypernym, Engine), and (BMW_Z3, configuration, Diesel_engine) add a value to document d_e, too. The engine of all BMW Z3 models requires fuel and there exists no original BMW Z3 model with a diesel engine. Associative SCD (BMW_Z3, configuration, Diesel_engine) is wrong if we think about available Z3 configurations having an original engine. However, enriching SCD set $T(d_e)$ with SCD (BMW_Z3,configuration, Diesel_engine) adds value to the task of an agent offering a document retrieval service, since people searching for the Z3 model containing a diesel engine might receive document *doc_ 7071* because of the new SCD. It follows that those people and can directly extract the information from the document's content that there exist no original Z3 model with a diesel engine.

BMW Series 1 The content of a second document d'_e (*doc_ 3407*) describes the Series 1 from BMW. Some models of this Series have a diesel engine. The authors of document d'_e mention three times the word *diesel* in the text but the corresponding DBpedia KB does not contain SCD (BMW_1_Series, configuration, Diesel_engine). However, Alg. 5 enriches $T(d'_e)$ with the associative SCD that is again beneficial for an application like document retrieval and query answering.

BMW Roadsters Algorithm 5 enriches (BMW_1_Series, subject, Category: Roadsters) to the SCD associated with *doc_ 3407*. This SCD only imprecisely describes the BMW 1 series. All cars from this Series have four seats and some of the cars are convertible. A short definition of a roadster is the following: *open two-seat cars of sporting appearance or character.* We assume that Alg. 5 enriches $T(d'_e)$ with this SCD, because the text in d_e is similar to some other documents being from category roadsters. There are models in this series which are convertibles and the associative SCD (BMW_1_Series, subject, Category: Roadsters) might be useful for people searching for convertibles from BMW, because many people use roadster and convertible interchangeable.

BMW M41 Engine The content of document *doc_ 3407* describes the M41 injection Diesel engine produced from 1994 through 2000. Obviously, an engine is no hypernym for a car and is no vehicle. Thus, both SCDs associated with *doc_ 3407* represent false positive SCDs.

Chapter 6

Identifying Subjective Content Descriptions within Texts

So far, we have assumed that SCDs and documents are separate or at least clearly distinguishable. However, an agent in pursuit of new documents may come across documents where normal document text, i.e., content, and textual content descriptions, i.e., SCDs, are interleaved. An agent could go through the text and identify the SCDs manually or create a parser that separates the SCDs from the content by specifying rules for distinguishing content and SCDs. However, both approaches are cost-intensive and laborious, and require intimate knowledge about content and SCDs of documents.

A special class of documents are represented by *historical-critical editions* representing an addition to the original text of documents, where (different) philologists add explanations, e.g., add data about the original text or data somehow related to the text. A historical-critical edition presents text that is as authentic as possible and has been cleared of errors and the text represent original content interleaved by textual content descriptions.

Let us assume that an author has written a poem in the 1^{st} century after Christ. Different philologists might have added explanations to the original poem to explain the genesis of the text. So, one philologist might have added some explanations in the 6^{th} century, another philologists might have added further explanations in the 14^{th} century, and further explanations might have been added in the 19^{th} century by two other philologist. A historical-critical edition is not only a combination of all explanations but represents a reliable basis for academic study of the original text cleared of errors (Plachta (1997)).

An agent with knowledge about the original poem can distinguish poem and comments within a historical-critical edition. The first problem we tackle in this chapter is that of automatically identifying inline SCDs (iSCDs) within texts. We turn to the agent's corpus containing documents to solve the problem of identifying inline SCDs within texts. Assuming that the unknown document with iSCDs is of the same context as the documents in the corpus, we can use the corpus-specific SCDs-word probability distribution to distinguish between content and SCDs. The SCD-word probability distribution allows for computing most probably suited MPSCDs for the unknown document based on the

similarity between words in a document and words usually occurring with an SCD. If the words belong to the content, we expect that an agent can identify a corresponding MPSCD with a high similarity value. If the words belong to an SCD, which we assume has a different composition of words occurring together, we expect that the similarity value of the MPSCD is low. Based on these expectations, we set up an HMM where the hidden state variable encodes if a word sequence belongs to content or SCDs and the observation sequence consists of discretized similarity values. Given this setup and the existing corpus, the agent can train an HMM and then compute a most-likely sequence of hidden states using the Viterbi algorithm to identify the iSCDs within texts. Once iSCDs are identified, the agent can use them for further purposes, e.g., translating content and comments.

The second problem we tackle in this chapter is that of context-specific translation of content and SCDs for humanities working with historical-critical edition. Different dictionaries are necessary to extract the sense of a word, since the sense of a word might change over time and content and SCDs are often from different centuries. Assuming that an agent has translations for a set of documents, we can create dictionaries s.t. each dictionary contains for each word exactly one context-specific translation. For a new document with content and iSCDs identified, we present a context-specific dictionary selection approach based on n-grams, i.e., sequences of n neighboring words, to identify the best suited dictionary for the content and for textual content descriptions.

Specifically, the contributions of this chapter are:

(i) an approach to identifying iSCDs within texts based on an HMM, and

(ii) an approach to identifying a suitable dictionary for content and iSCDs.

Additionally, we present a case study based on real-world and simulated data to evaluate the performance of both approaches we introduce in this chapter.

6.1 Identifying Inline SCDs

This section introduces the problem of identifying textual SCDs that are interleaved with content text in a document and presents an HMM-based approach solving the problem.

We refer to SCDs that are interleaved with the text of a document as iSCDs and to the remaining text as *content*. Given this new setting, we introduce iSCDs into our notation.

- An iSCD t is a sequence of words $(s_1, \ldots, s_n), n \in \mathbb{N}, s_i \in \mathcal{V}_{T(\mathcal{D})}$ that is associated with the sequence of words exactly preceding t in d.
- Next to the vocabulary $\mathcal{V}_{\mathcal{D}}$ of corpus $\mathcal{D}$, there is a vocabulary $\mathcal{V}_{T(\mathcal{D})}$ of the words occurring in the (inline) SCDs t in $T(\mathcal{D})$. The words of both vocabularies may overlap.

- A document d is represented as a sequence of words from $\mathcal{V}_{\mathcal{D}}$ and $\mathcal{V}_{T(\mathcal{D})}$ with subsequences of words from $\mathcal{V}_{\mathcal{D}}$ and subsequences of words from $\mathcal{V}_{T(\mathcal{D})}$ alternating, where the latter is associated with the preceding window of words. Further SCDs may be located throughout d.

6.1.1 Inline SCD Detection Problem

The problem at hand consists of an agent being faced with a document containing content in the form of text and textual SCDs and no markers or way inherently available to distinguish the two. An important task of an agent is to identify iSCDs within text s.t. the agent can (i) reconstruct the content of a document containing iSCDs, and (ii) use the located iSCDs, e.g., to identify similar documents within the corpus.

Problem 6.1.1 introduces the inline SCD problem and Example 6.1.1 illustrates the problem using a short text.

Problem 6.1.1 (Inline SCD Problem)**.** *An agent does not know which subsequences of words belong to content and which belong to iSCDs for a document* $d = (w_1^d, \ldots, w_D^d), w_i^d \in (\mathcal{V}_{\mathcal{D}} \cup \mathcal{V}_{T(\mathcal{D})})$.

Example 6.1.1 (Inline SCD Example)**.** Assume that a new document $d \notin \mathcal{D}$ contains the following sentence with two iSCDs:

> "David Blei professor at Columbia University received the ACM Infosys Foundation Award renamed in the ACM Prize in Computing in 2013."

For illustration purpose, the two iSCDs are underlined here. Note that the highlighting is not available in the original document. An agent is faced only with the word sequence and has to decide which words represent iSCDs. The problem is simple if the two vocabularies for iSCDs and content do not overlap but the problem becomes increasingly harder the more the two vocabularies share words and those words have similar frequencies regarding occurrences.

We can formalize the inline SCD detection problem as a classification problem namely estimating the corresponding category for each word in a sequence of words. In our setting, we have two categories. One category represents the subsequences in d that are not part of an iSCD, i.e., content, and another category represents the subsequences in d belonging to an iSCD.

6.1.2 General Procedure

To solve the inline SCD detection problem introduced in Problem 6.1.1, we work with two assumptions about corpus $\mathcal{D}$ and a new document d containing content text and iSCDs:

(i) Document d belongs to the same context as $\mathcal{D}$.

(ii) Vocabulary $\mathcal{V}_{\mathcal{D}}$ differs at least slightly from $\mathcal{V}_{T(\mathcal{D})}$.

Given the first assumption, we can use SCD-word probability distribution $\delta(\mathcal{D})$ for d as well. Given the second assumption, $\delta(\mathcal{D})$ works well for estimating MPSCDs for content words and less well for iSCDs words. Estimating MPSCDs for d using a sliding window, we expect the following behavior: The MPSCD similarity value for a window over an iSCD should be significantly smaller than the MPSCD similarity value for a window over content. We can train an HMM with this behavior to solve Problem 6.1.1. Specifically, we need to perform the following steps:

(i) Estimate $\delta(\mathcal{D})$ for $\mathcal{D}$ using Alg. 1 (offline).

(ii) Train an HMM for classifying whether a word belongs to the category "content" or the category "iSCD" (offline).

(iii) For each new document d:

  (a) Estimate the MPSCD sequence over a sliding window along the words of d using $\delta(\mathcal{D})$.

  (b) Compute the most probable sequence of states S in the HMM given the sequence of similarity values of the MPSCD sequence as evidence and identify the words in d that belong to category "iSCD" given S.

The following sections present in detail how to estimate the MPSCD similarity value sequence and classify iSCDs using an HMM.

6.1.3 Estimating a Sequence of MPSCD similarity values

Given a corpus $\mathcal{D}$ containing documents associated with a set of location-specific SCDs, we can generate a corpus-specific SCD-word probability distribution $\delta(\mathcal{D})$ using Alg. 1. Based on $\delta(\mathcal{D})$, Alg. 6 allows for calculating a sequence of MPSCDs similarity values for d by sliding a window ($win_{d,\rho}$) of size σ over the words in d. Initially, the sliding window contains the first σ words $(w_1^d, ..., w_\sigma^d)$. Then, we shift the window over the sequence of words in d by removing the first word (w_1^d) and extending the window with the word $w_{\sigma+1}^d$. We repeat this shifting operation until the end of the document, with the last window containing $(w_{(D-\sigma)}^d, ...w_D^d)$. For the sequence of words in each $win_{d,\rho}$, Alg. 6 calculates the MPSCD t and corresponding similarity value sim. Specifically, Alg. 6 builds a vector representation $\delta(win_{d,\rho})$ for the sequence of words in $win_{d,\rho}$ and compares the vector with the vector representation of all SCDs in $\delta(\mathcal{D})$ using the cosine similarity. Example 6.1.2 describes the sliding window behaviour and what similarity values might look like using an example setup.

Algorithm 6 Estimating MPSCD sequence

function ESTIMATEMPSCDSEQUENCE(d, σ, $\delta(\mathcal{D})$)
  Input: Document d, window size σ, SCD-word distribution $\delta(\mathcal{D})$
  Output: $\mathcal{W}$: Triple set containing SCDs (t) with similarity values (sim) and window position ($win_{d,\rho}$)
  $\mathcal{W} \leftarrow \emptyset$
  for $\rho \leftarrow \frac{\sigma}{2}$; $\rho \leq \#word(d)$; $\rho \leftarrow \rho + 1$ **do**
    Set up $win_{d,\rho}$ of size σ around ρ with $t = \perp$
    $\delta(win_{d,\rho}) \leftarrow$ new zero-vector of length V
    for each word $w \in win_{d,\rho}$ **do**
      $\delta(win_{d,\rho})[w]$ += $I(w, win_{d,\rho})$
    $t \leftarrow \arg\max_i \frac{\delta(\mathcal{D})[i] \cdot \delta(win_{d,\rho})}{|\delta(\mathcal{D})[i]| \cdot |\delta(win_{d,\rho})|}$ in $win_{d,\rho}$
    $sim \leftarrow \max_i \frac{\delta(\mathcal{D})[i] \cdot \delta(win_{d,\rho})}{|\delta(\mathcal{D})[i]| \cdot |\delta(win_{d,\rho})|}$
    $\mathcal{W} \leftarrow \mathcal{W} \cup (t, sim, win_{d,\rho})$
  return $\mathcal{W}$

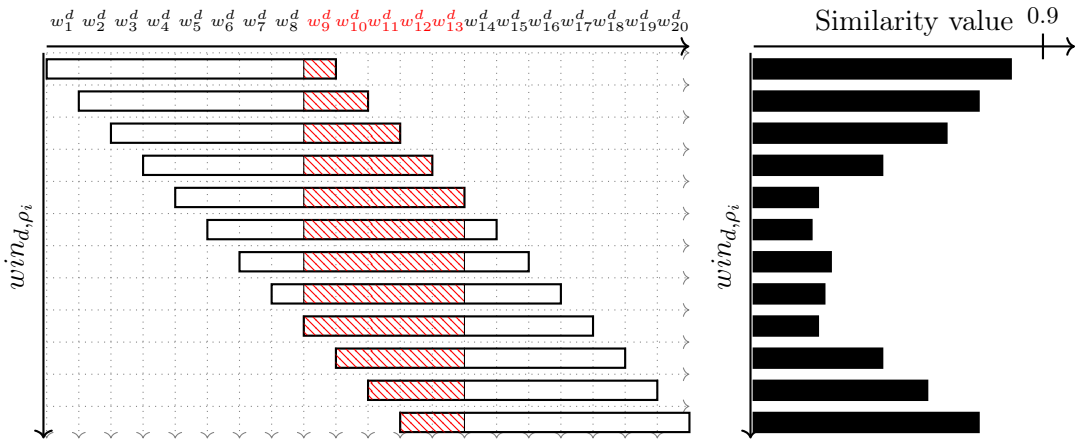


Figure 6.1: Sliding window of size $\sigma = 9$ sliding over document d. Words w_9^d to w_{13}^d represent an iSCD, accompanied by corresponding similarity values on the right.

Example 6.1.2 (Sliding Window and Similarity Values)**.** Figure 6.1 illustrates the behavior of a sliding window $win_{d,\rho}$ for the first 20 words w_1^d to w_{20}^d in d with a window size σ of 9, starting with window $win_{d,w_5^d} = (w_1^d, ..., w_9^d)$ and ending with window $win_{d,w_{16}^d} = (w_{12}^d, ..., w_{20}^d)$. In this example, the words w_9 to w_{13} represent an iSCD and are colored red. Sliding the window from left to right the number of words corresponding to content and SCD changes.

Figure 6.1 also shows the sequence of MPSCD similarity values on the right (black bars) for each window depicted to the left. We have a sequence of twelve MPSCD similarity

values for the corresponding twelve windows, given by:

$$(0.8, 0.7, 0.6, 0.4, 0.2, 0.18, 0.24, 0.22, 0.22, 0.4, 0.54, 0.7)$$

In the first window win_{d,w_5^d}, only word w_9 belongs to an iSCD. Shifting the window to the right results in a second word belonging to the iSCD. The more words belong to an iSCD, the lower the corresponding window's MPSCD similarity value gets. The reason why the MPSCD similarity value gets lower the more words belong to an iSCD is the difference in the vocabularies for content and iSCDs.

As shown in Fig. 6.1, iSCDs yield a specific pattern in the sequence of MPSCD similarity values: Similarity values first decrease, when the sliding window slowly moves into the iSCD, plateau, when the sliding window is squarely in the iSCD, and then increase again, with the sliding window moving out of the iSCD. Next, we present an approach to solve Problem 6.1.1 by identifying iSCDs in d using an HMM trained of sequences of MPSCD similarity values.

6.1.4 Estimating iSCDs

We use an HMM to identify iSCDs using the sequence of MPSCD similarity values in d. Using a set of documents containing located SCDs, we can calculate MPSCDs and their corresponding similarity values to train an HMM on this information using the Baum-Welch algorithm. The discrete observation alphabet Δ requires discretizing similarity values. A discretization function $f : [0, 1] \mapsto \Delta$ maps a MPSCD similarity value (x) to one of the m symbols in Δ. The specific discretization depends on the agent's task and can be adapted to each problem individually.

To solve Problem 6.1.1, we have to find the most likely sequence of states in an HMM λ, given a sequence of MPSCD similarity values $\mathcal{W}$. Algorithm 7 describes the workflow for identifying iSCDs based on such a sequence of similarity values $\mathcal{W}$ over the windows in d. First, Alg. 7 calculates the most likely state sequence by applying the discretization function f on the similarity values in $\mathcal{W}$, yielding an observation sequence O for λ. Then, it calculates the most likely sequence of states S in λ given O using the Viterbi algorithm, which makes use of the dynamic programming trellis for computing the most likely state sequence S for an observation sequence O. Given S, Alg. 7 reconstructs the iSCDs by identifying the windows behind those states in S that are equal to s_2 ("iSCD") and marking the words in each corresponding window as an iSCD. Example 6.1.3 illustrates estimating the most likely sequence of states using the Viterbi algorithm for MPSCD similarity values resulting from the twelve windows shown in Example 6.1.2.

Example 6.1.3. Let us assume that Alg. 6 yields the following MPSCD similarity values for the 12 windows in Example 6.1.2, as shown in Fig. 6.1:

$$(0.8, 0.7, 0.6, 0.4, 0.2, 0.18, 0.24, 0.22, 0.22, 0.4, 0.54, 0.7)$$

Algorithm 7 Estimating iSCDs using MPSCD similarity values and a trained HMM

```
1: function estimateISCDs(λ, f, 𝒲, d, σ)
2:     Input: HMM λ, discretization function f, similarity values 𝒲, document d,
           window size σ
3:     Output: SCDs T(d)
4:     O ← ()
5:     for each similarity value sim ∈ 𝒲 do
6:         O ← O ∘ f(sim)
7:     S ← Viterbi(λ, O)                    ▷ Compute the most likely state sequence
8:     T(d) ← ∅
9:     for each i, state ∈ S do             ▷ Iterate over S, maintain an index i
10:        if state = s_2 then              ▷ s_2 corresponds to "iSCD"
11:            t ← (w_i^d, ..., w_{i+σ}^d)
12:            T(d) ← T(d) ∪ t
13:    return T(d)
```

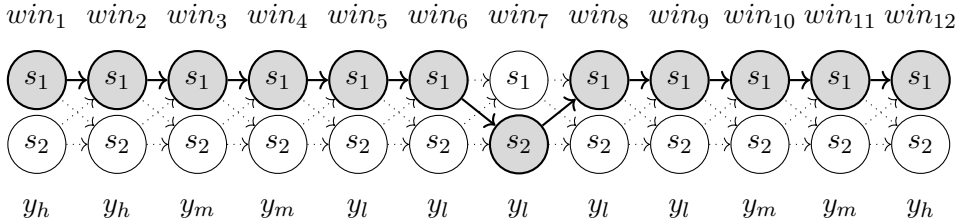


Figure 6.2: Trellis corresponding to the MPSCD sequence of Example 6.1.2.

Using the following function for discretization

$$f(x) = \begin{cases} y_l & 0 \leq x < \theta_1 \\ y_m & \theta_1 \leq x < \theta_2 \\ y_h & \theta_2 \leq x \leq 1, \end{cases} \tag{6.1}$$

where $\theta_1 = 0.3$ and $\theta_2 = 0.7$, we get the following observation sequence:

$$O = (y_h, y_h, y_m, y_m, y_l, y_l, y_l, y_l, y_l, y_m, y_m, y_h)$$

Let the corresponding state sequence in the trained HMM be given by

$$S = (s_1, s_1, s_1, s_1, s_1, s_1, s_2, s_1, s_1, s_1, s_1, s_1).$$

Figure 6.2 represents the trellis of the observation sequence O, where the thick arrows indicate the most probable transitions between the states and the dotted lines represent

all possible state transitions. The hidden iSCD is at position win_7, which corresponds to window $win_{d,\rho}$ with $\rho = 7 + \lfloor \frac{9}{2} \rfloor = 11$. The identified window contains the words $(w_9^d, \dots w_{13}^d)$, which make up the iSCD in the example.

Correctness By identifying the word sequences that are most probably iSCDs, we automatically identify which words belong to iSCDs and which words belong to content. Therefore, we solve Problem 6.1.1 by not only providing which subsequences are iSCDs but providing those that are most probable given the underlying HMM, which makes the quality of the solution to a specific instance of the problem optimal in the sense that given the probabilistic fundamentals of our approach, this calculated solution is the one leading to the highest probability.

6.2 Context-specific Dictionary Selection

In the last section, we have presented an HMM-based approach to identifying iSCDs within texts. In this section, we introduce the problem of selecting a suitable dictionary to translate both, iSCDs and content. Additionally, we present an approach to solve the problem by automatically selecting suitable dictionaries.

6.2.1 Dictionary Selection Problem

A person interested in translating words in a document has to decide on a dictionary for unknown words to select a suitable translation. Generally, there is no single translation for a word since the translation of a word depends on the specific context. Again, let us consider historical-critical editions representing poems as content and iSCDs as comments. Translating text from a historical-critical edition is non-trivial since the edition contains text written from different authors at different times. So, using a single dictionary to translate the text of a historical-critical edition might lead to non-optimal translations because a single dictionary ignores the fact that a poem has been enriched with additional descriptions at different points in time. Thus, an agent might be interested in selecting the k best suited dictionaries for a poem and each identified iSCD, s.t. e.g., humanities can the best suited dictionaries to translate the document.

Generally, the dictionary selection problem asks for the top-k dictionaries for a sequence of words $(w_1, \dots, w_j)$ representing the poem or an iSCD, given J corpora $\{\mathcal{D}_i\}_{i=1}^J$ each containing a set of documents and their corresponding translations. Mathematically, we can describe the problem as follows:

$$\underset{dict_1,\dots,dict_k \in Dict}{\arg\max} \quad P(dict_1, \dots, dict_k | (w_1, \dots, w_n), \{\mathcal{D}_i\}_{i=1}^J) \tag{6.2}$$

Algorithm 8 N-gram-based Dictionary Selection

function SELECTDICTIONARY(n, $\{\mathcal{D}_i\}_{i=1}^{J}$, seq)
  Input: length of n-gram n, set of corpora $\{\mathcal{D}\}_i^J$, word sequence seq
  Output: suitable dictionary: $dict$
  $v_{seq} \leftarrow$ n-gram-frequency of seq
  $s \leftarrow 0$
  $dict \leftarrow$ empty
  for each corpus $\mathcal{D} \in \{\mathcal{D}_i\}_{i=1}^{J}$ **do**
    $v_{\mathcal{D}} \leftarrow$ corpus-specific n-gram-frequency
    $sim \leftarrow \frac{v_{\mathcal{D}} \cdot v_{seq}}{|v_{\mathcal{D}}| \cdot |v_{seq}|}$
    if $sim > s$ **then**
      $dict \leftarrow dict(\mathcal{D})$
      $s \leftarrow sim$
  return $dict$

6.2.2 Dictionary Selection Approach

A sequence of words in a historical-critical edition might represent the original poem or an iSCD. Generally, for a word different translations exist and for a human a suitable translation is only possible if the corresponding context is available. For a new document containing only words and no SCDs, we can represent the context of each word by its neighboring words. This section presents an approach to identify the most suitable dictionaries to translate a word sequence available in d based on the n-grams occurring within the documents of the J corpora and the n-grams in an iSCD. Thus, for each word in a sequence of words we create the n-grams by using the neighboring words. Afterwards, we identify a suitable dictionary for a sequence of words based on the similarity of all sequence-specific n-grams and all n-grams one can generate from documents of all J corpora. Finally, the dictionary associated with the corpus containing the most similar n-grams is selected. Let us assume we have a set of J corpora and for each corpus $\mathcal{D}$ exists a corpus-specific dictionary that has been used to translate the documents, e.g., dictionary $dict_1$ is suitable for documents in $\mathcal{D}_1$ and another dictionary $dict_2$ suitable for documents in corpus $\mathcal{D}_2$, and so on. Furthermore, we assume a new document d is available and Alg. 7 has successfully identified the content and comments. Then, we are interested in selecting a suitable dictionary for content in d and a suitable dictionary for each iSCD based on the n-grams from the sequences of words referring to the text and to the iSCDs.

Algorithm 8 represents the technical description for selecting a suitable dictionary for a word sequence seq. The inputs are the length n of n-grams, all corpora $\{\mathcal{D}\}_{\rangle i=1}^{J}$, and a word sequence seq. The output of Alg. 8 is the best suited dictionary $dict$. Algorithm 8 starts by calculating the frequency v_{seq} of n-grams available in word sequence seq. Then, Alg. 8 calculates the n-gram frequencies $v_{\mathcal{D}}$ for each corpus $\mathcal{D}$ by analyzing each document

in $\mathcal{D}$ and computes the cosine similarity between v_{seq} and $v_{\mathcal{D}}$. Alg. 8 stores the current corpus' dictionary if it has a higher similarity value as any corpus checked before. In the end, Alg. 8 returns the stored dictionary.

Correctness We calculate all possible n-grams from a given sequence of words representing an iSCD or a poem. Additionally, we calculate all possible n-grams for each document in the J corpora. Thus, we can compare the n-gram frequency vectors from a sequence of words of new document d with each corresponding n-gram frequency vector of each corpus. Using a vector representation of the n-gram frequency allows to compare the frequencies based on the cosine similarity used in line 9 of Alg. 8. Comparing the n-gram frequency vector of a word sequence with the n-gram frequency vectors of each corpus yields the most similar corpus. Since each corpus is linked to a corpus-specific dictionary, we can directly select the best suited dictionary for each sequence of words in d.

6.3 Case Study

After introducing the HMM-based approach to identifying iSCDs within texts, followed by the n-gram-based dictionary selection approach, we present a case study illustrating the potential of both approaches estimating iSCDs within documents for different data sets as well as selecting suitable dictionaries. We start with a description of data sets we use in this case study followed by the workflows for analyzing the performance of the HMM-based iSCDs detection approach and the n-gram-based dictionary selection. Additionally, we illustrate the potential of both approaches by evaluating their performance on the different data sets.

6.3.1 iSCD Detection

This section presents data sets, the workflow and results for the iSCD detection approach.

Data Sets

We use the following seven data sets to evaluate the performance of the HMM-based approach to identifying iSCDs in a new document.

1. `Tamil` consists of 91 poems transcribed from old palm leaves (Wilden (2018)).
2. `Greek` consists of 1 treatise about Aristotle's Categories (Bekker and others (1831)).
3. `US` consists of 74 articles about cities in the Unites States of America[1].

[1] US cities – `https://en.wikipedia.org/wiki/List_of_United_States_cities_by_population`

4. `EU` consists of 10 articles about cities in Europe[2].

5. `Arxiv-general` consists of 500 randomly selected abstracts from Arxiv[3].

6. `Arxiv-CS` consists of 500 randomly selected abstracts from publications of the Computer Science (CS) category available on Arxiv[3].

7. `Newsgroups` consists of 290 posts from the 20 newsgroups data set containing 18.828 newsgroup posts on 20 topics [4].

For the `Tamil` data set, we extract the documents from Wilden (2018). For the `Greek` data set, we extract the documents from the Alignment of Aristotle's Categories[5] containing a digital copy of the Aristotle's Categories. Each document is associated with comments about the content from the original documents and acting as iSCDs. The `US` and `EU` data sets consist of articles from the open and widely accessible online encyclopedia *Wikipedia* containing text about cities in the US and EU, respectively.

The `Arxiv` data set contains a subset of 500 randomly selected abstracts from the Arxiv library. Additionally, for the `Arxiv-CS` data set we randomly selected 500 abstracts from the Arxiv library referring to the CS category. The `Newsgroups` data set represents a randomly selected subset of the well-known 20 newsgroups data set.

In contrast to the documents in `Tamil` and `Greek`, the documents in the other data sets do not contain iSCDs. Thus, we generate iSCDs for documents in data sets (3) - (7) by (i) downloading a dump of the English free online dictionary *Wiktionary*[6], (ii) creating an SCD for each word in a document using the corresponding Wiktionary entry (if available), and (iii) splicing the SCDs into the document.

After we have downloaded all documents and determined iSCDs we store the documents in the respective corpora. Next, we use the following standard preprocessing tasks from the NLP community, (i) lowercasing all characters, (ii) stemming the words, (iii) tokenizing the result, and (iv) eliminating tokens from a stop word list to transform the text of the documents into a more digestible form for machine learning algorithms to increase their performance.

Generally, language changes over time, and modern word stemmers and stop words might be non-optimal for old texts. Thus, we neither eliminate stop words nor stem words in `Greek`. For `Tamil`, we compare the performance of the unstemmed (unst.) corpus to a stemmed (st.) corpus by a modern word stemmer[7] for Tamil.

Table 6.1 gives an overview about

[2]European cities – `https://en.wikipedia.org/wiki/List_of_urban_areas_in_Europe`
[3]ArXiv – `https://www.kaggle.com/Cornell-University/arxiv`
[4]20 Newsgroups `http://qwone.com/~jason/20Newsgroups/`
[5]Aristotle's Categories – `http://textalign.net/output/ar.cat.tan-a-div-collated-obj.html`
[6]`https://www.wiktionary.org`
[7]`https://github.com/rdamodharan/tamil-stemmer`

Table 6.1: Characteristics of data sets divided into eight different settings.

	Tamil		Greek	US	EU	Arxiv		News-
	Unst.	St.				general	CS	groups
$\lvert\mathcal{D}\rvert$	91	91	1	74	10	500	500	290
Avg $\#word(d)$	73,2	73,2	10.458	200,7	318,7	74,4	77,3	133,7
# iSCDs	908	869	143	1.814	757	4.905	4.715	6.541
$\lvert\mathcal{V}_{\mathcal{D}} \cup \mathcal{V}_{T(\mathcal{D})}\rvert$	8.015	5.783	3.623	6.688	3.314	10.083	8.843	12.829
$\lvert\mathcal{V}_{\mathcal{D}}\rvert$	5.521	4.304	1.894	3.657	1.439	5.776	4.715	7.558
$\lvert\mathcal{V}_{T(\mathcal{D})}\rvert$	2.666	1.987	2.123	3.902	2.232	6.751	6.493	8.332
$\lvert\mathcal{V}_{\mathcal{D}} \cap \mathcal{V}_{T(\mathcal{D})}\rvert$	172	508	394	871	357	2.444	2.365	3.061

(i) the number of documents in a corpus,

(ii) the average length of documents, and

(iii) the size of different vocabularies.

Stemming the poems in `Tamil` yields to less different words in the vocabulary and more intersecting words between content (poem) and their comments (iSCDs). For ease of reading, we divide numbers with many digits into groups using a dot as delimiter.

iSCD Detection Workflow

Evaluating the performance of the HMM-based approach to identifying SCDs within texts requires a setup of the HMM. We use two hidden states (s_1,s_2) as defined in Definition 4.3.1 and five observable states y_1 to y_5. We have tested different setups and five observable symbols having yielded good results. The thresholds are selected by analyzing histograms of MPSCD similarity values. Figure 6.3 represents three histograms of MPSCD similarity values gained from Alg. 6 for the three corpora. The intervals for y_1 to y_5 should be selected from the areas of similarity values having a high frequency to gain a good HMM-based iSCD detection performance. For all data sets expect `Greek`, we select the intervals: $y_1 = [0.0, 0.05)$, $y_2 = [0.05, 0.1)$, $y_3 = [0.1, 0.15)$, $y_4 = [0.15, 0.20)$, and $y_5 = [0.20, 1.0]$. For `Greek`, we use the following intervals: $y_1 = [0.0, 0.3)$, $y_2 = [0.3, 0.35)$, $y_3 = [0.35, 0.4)$, $y_4 = [0.4, 0.45)$, and $y_5 = [0.45, 1.0]$.

We perform the following five tasks on each data set to evaluate the performance of the HMM-based iSCD detection approach:

(i) Split the data set into multiple parts and use leave-one-out cross-validation resulting in training sets of 80-90% of documents and test sets of remaining 10-20% of documents.

(ii) Form the SCD-word probability distribution δ for the training set using Alg. 1.

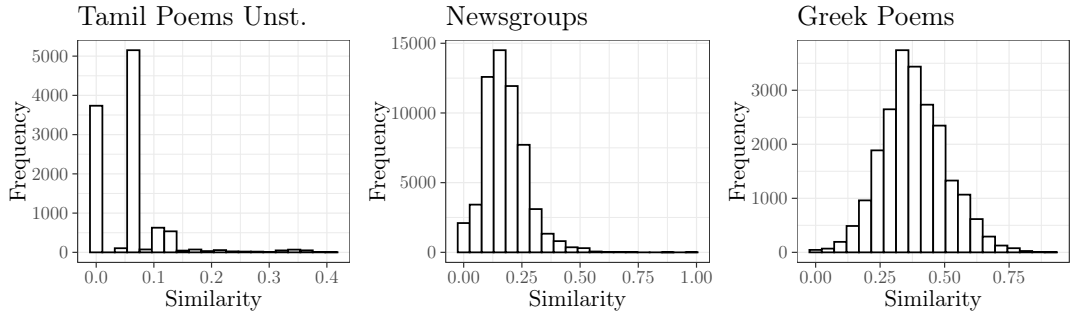


Figure 6.3: Histograms of MPSCD similarity values gained from Alg. 6 for unstemmed `Tamil` (left), `Newsgroups` (center), and `Greek` (right). *Note the different scaling of the axes.*

(iii) Generate an HMM and apply the Baum-Welch algorithm to train the hidden parameters of the HMM.

(iv) For each document in the test set estimate MPSCDs using Alg. 6, and

(v) Compute the most probable sequence of hidden states in the HMM for the discretized sequence of similarity values using the Viterbi algorithm.

We use cross-validation and the total average of the precision, recall and F_1-score in the results.

We evaluate the performance of the HMM-based approach by comparing the results with the following two standard approaches, namely, word-based classification and a single threshold-based classification.

Word-based Classification For each word, we form a probability distribution representing how often the word occurs in content ($\#_c$) vs. iSCD ($\#_s$), i.e., $p = \frac{\#_c}{\#_c+\#_s}$ and $1-p$. We classify each word by sampling from $(p, 1-p)$. Words belonging only to one vocabulary have a $(1, 0)$ probability distribution and can be directly classified as belonging to either content or iSCD. For words that are not part of any vocabulary, we randomly assign a category.

Threshold-based Classification Instead of training an HMM on the MPSCD similarity sequences, we directly classify based on the MPSCD similarity value of a window sim and a threshold ℓ. If $sim < \ell$, we classify the words in the window as an iSCD. We use the histograms in Fig. 6.3 to choose the values of ℓ. For unstemmed `Tamil`, $\ell = 0.05$ and for `Greek`, $\ell = 0.35$ results in best performance for iSCD detection. For all other data sets, $\ell = 0.1$ yields the best results.

As depicted in Table 6.1, unstemmed `Tamil` contains only 172 words, representing 2% of all words, that occur in both vocabularies $\mathcal{V}_\mathcal{D}$ and $\mathcal{V}_{T(\mathcal{D})}$. After stemming the words

in `Tamil`, this share increases to 8%. In `US`, `EU` and `Greek` $\mathcal{V}_{\mathcal{D}}$ and $\mathcal{V}_{T(\mathcal{D})}$ share around 10% of their words,respectively, while `Arxiv` and `Newsgroups` share around 25% of their words, respectively. Thus, we expect a good word-based classification performance for data sets sharing less words between $\mathcal{V}_{\mathcal{D}}$ and $\mathcal{V}_{T(\mathcal{D})}$, e.g., for `Tamil`.

Next, we present the results for the HMM-based iSCD detection approach.

Results

Figure 6.4 presents the performance of the HMM-based, word-based, and threshold-based approach for all data sets. For the HMM-based approach, we present the performance of the initial model and the trained model. The initial HMM contains the following emission probabilities in case of iSCDs:

$$\{y_1 : 0.15, y_2 : 0.50, y_3 : 0.20, y_4 : 0.10, y_5 : 0.05\}$$

and in case of text:

$$\{y_1 : 0.05, y_2 : 0.05, y_3 : 0.30, y_4 : 0.35, y_5 : 0.25\}.$$

Unstemmed `Tamil` yields very low MPSCD similarity values (Fig. 6.3) in contrast to the other data sets yielding values similar to `Newsgroups`. Therefore, we slightly modify the emission probabilities for unstemmed `Tamil` by increasing y_1 and y_2. The emission probabilities encode that a window associated with an MPSCD of a high similarity value is unlikely being classified as an iSCD.

The word-based classification yields the best F_1-Score for `US`. The precision of the word-based classification is very high for `Tamil` and `Greek`, while the precision is low for `Arxiv`. This corresponds to our assumption made before: Less shared words between $\mathcal{V}_{\mathcal{D}}$ and $\mathcal{V}_{T(\mathcal{D})}$ lead to better precision. The recall of the word-based classification is considerably lower than the recall of the HMM-based approach. Interestingly, the word-based classification performance is not as poor as expected for data sets containing more overlapping vocabulary between $\mathcal{V}_{\mathcal{D}}$ and $\mathcal{V}_{T(\mathcal{D})}$.

The threshold-based classification performance is good for all data sets. Often the results are similar or even better than the results of the HMM-based approach. However, for the threshold-based approach it is difficult to determine the best values ℓ for the threshold, since the threshold changes for each data set.

The HMM-based approach performs well for all data sets. Mostly, the trained HMM, the initial HMM, and the threshold-based classification result in similar values. The difference between the initial and the trained HMM is not as pronounced since the initial values for the emission probabilities are already of good quality, which reduces not only the runtime for learning but also has the effect of the initial model performing relatively well. The discretization function uses intervals close to threshold ℓ, such that threshold-based classification performs similar well, too.

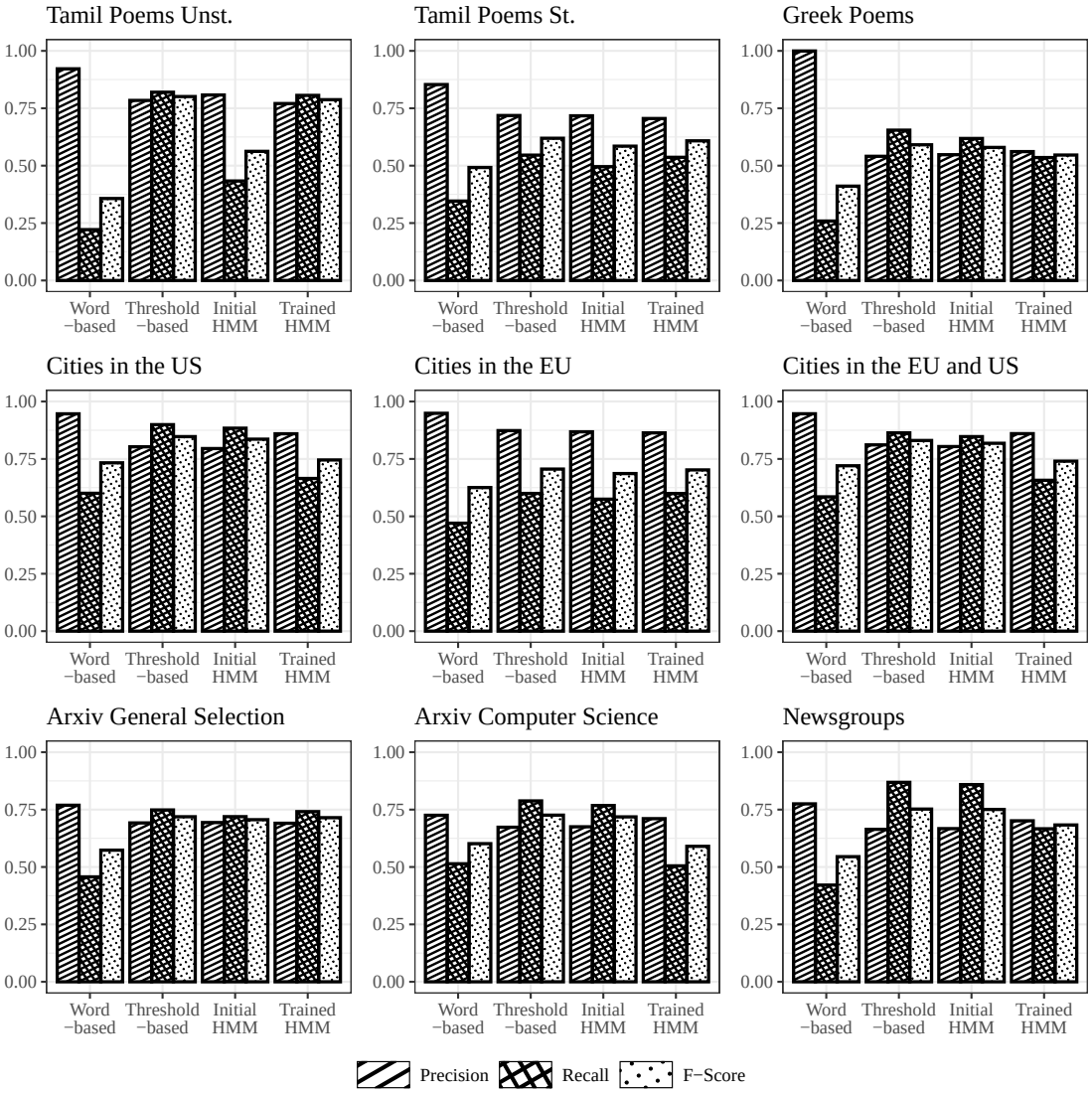


Figure 6.4: Classification performance for HMM-, word-, and threshold-based approach.

Overall, the HMM-based approach yields the best performance in the case study. Even thought, the threshold-based classification sometimes results in slightly better results for most data sets. However, choosing a threshold is difficult and training an HMM is much easier.

6.3.2 Dictionary Selection

This section presents the data sets, the workflow and results for the dictionary selection approach.

Data Sets

For the evaluation of the dictionary selection approach, we use the entire 20 Newsgroups data set mentioned above, containing 18.828 documents. On average each document contains 16 sentences and each newsgroup features 4.942 unique words while sharing 8.639 words with other newsgroups.

Additionally, we use a data set (`Shakespeare`) containing 39 Shakespeare documents taken from *Project Gutenberg*[8]. On average, each document in `Shakespeare` consists of 2.526 sentences. The total number of different words in the vocabulary of both data sets is 139.788, while both data sets share only 9.183 words. 9.826 words occur only in `Shakespeare`, and 120.779 words occur only in documents of the 20 newsgroups data set.

For documents in both data sets, we apply the same four preprocessing tasks from the NLP community already used for detecting the iSCDs, namely lowercasing all characters, stemming the words, tokenizing the result, and eliminating tokens from a stop word list.

Dictionary Selection Workflow

Given multiple dictionaries and for each dictionary a well suited set of documents, the dictionary selection problem asks for the best suited dictionaries to translate a document d. In our scenario d is a poem interleaved with iSCDs. After extracting the content (d_p) and the set of iSCDs ($T(d)$) using the HMM-based iSCD detection approach, we perform the following two steps to translate the content in d:

(i) Specify the length of n-grams we use in Alg. 8.

(ii) Perform Alg. 8 to identifying a suitable dictionary for content and iSCDs.

We run all experiments on a virtual machine featuring 8 Intel 6248 cores at 2.50GHz (up to 3.90GHz) and 16GB RAM.

Using the two data sets we demonstrate the performance of the n-gram-based document selection approach in two different settings: First, we are interested in selecting the best

[8]https://www.gutenberg.org/

dictionary within a modern language dictionary for all 20 newsgroups and an ancient language dictionary for `Shakespeare`. Second, we assume to have a context-specific dictionary for each single newsgroup and we are interested in selecting the best dictionary within the 20 dictionaries.

For each setting we split the data sets randomly into a training set containing 80% of the sentences and a test set containing the remaining 20%. After training the models we measure the accuracy as proportion of correctly selected dictionaries for the sentences in the test set.

We compare the performance of the n-gram-based approach, introduced in Section 6.2.2, with the well-known Skip-gram model, introduced by Mikolov *et al.* (2013a,b). The training objective of the Skip-gram model is to find word representations that are useful for predicting the surrounding words in a sentence or a document. Given a sequence of training words, the objective of the Skip-gram model is to maximize the average log probability. We generate for each corpus a Skip-gram model and use distance measures to calculate the distance between a corpus-specific model and a given sentence from d to identify a suitable dictionary for a sentence based on the corpus where the corresponding model has the shortest distance. We analyze the performance of the following distance measures:

Jaccard Given a sentence, we take each word and query the model for the surrounding words. The words predicted are then compared to the actual words in the sentence using the Jaccard index. We use the mean of all Jaccard indices to get the distance between the sentence and a model.

Model Given a sentence of a document we calculate the distance of each word of the sentence to all words known by the model. To get the distance between the sentence and the model we use the mean of the distances for each word.

Sentence This approach is similar to *Model*, but we do not calculate the distance to all words known by the model, but only to the words occurring in the sentence. Thus, we also take each word of the given sentence and calculate the distance within the model to the other words in the given sentence.

Voting Calculate all three distance measures (Jaccard, Model, Sentence), predict the most probable model for each of the three measures and perform a majority voting. To determine the top-n best matching dictionaries, we predict the n most probable models for each distance measure and then perform a majority voting across the $3n$ predictions.

Results

Figure 6.5 shows boxplots of accuracies and ratio of predictable sentences. The ratio indicates for how many sentences the model could explicitly select a dictionary for, i.e.,

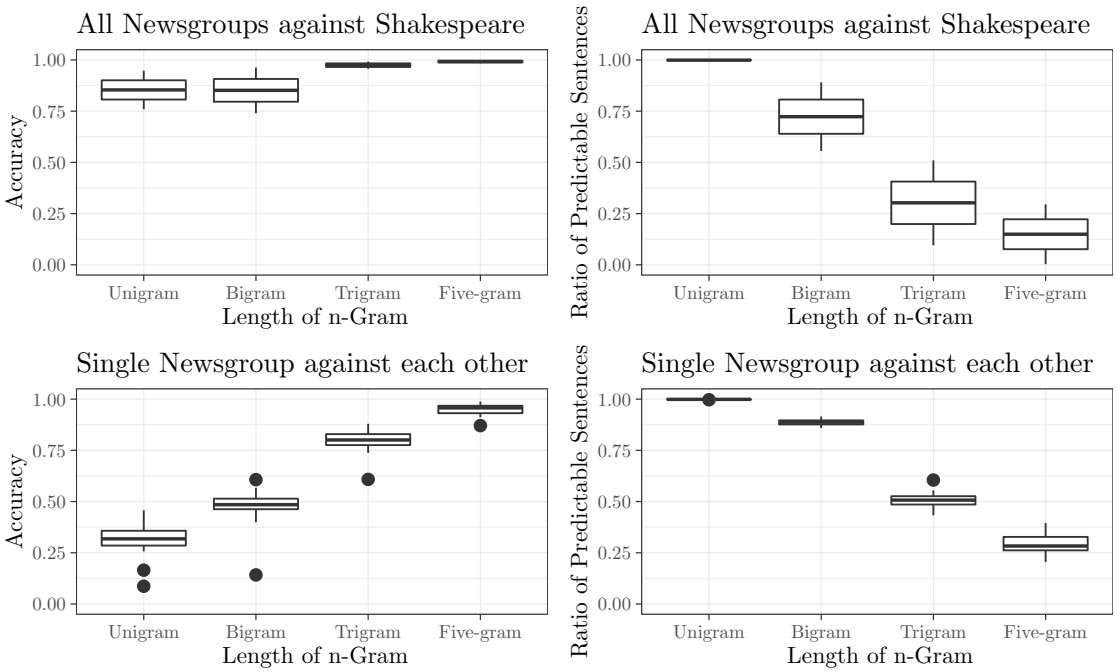


Figure 6.5: Accuracy (left) and ratio of predictable sentences (right) of the n-gram-based approach for different data sets and different lengths of n-grams.

when using five-grams it may happen that a sentence contains no five-gram known by the model and therefore no selection is possible.

In both settings, the model can predict all sentences using unigrams, but not while using five-grams. For `Shakespeare`, the accuracy is overall high, confirming that modern and ancient languages feature more differences than different contexts. For `Newsgroups`, the accuracy grows with the length of the n-grams while the ratio decreases. Generally, bigrams provide a good compromise between accuracy and ratio. Whilst rating the reached accuracy of 0.5 when using bigrams, we have to remember that randomly choosing a dictionary within 20 would result in an accuracy of around 0.05%.

As stated before, we compare the performance of the n-gram-based approach to the Skip-gram-based approach. In Fig. 6.6, we show the accuracy and ratio of the Skip-gram-based approach. The Skip-gram-based approach generally yields lower accuracy but provides higher ratios than the n-gram-based approach. For the Skip-gram-based approach, the `Sentence` distance measure results in the best accuracy. Comparing values for the best length of the n-grams with the best distance measure, bigrams and `Sentence` respectively, the accuracy of the n-gram-based approach is clearly higher while both approaches lead to similar ratios.

In Fig. 6.7, we illustrate the runtime of both approaches. On the left side, the performance of both approaches in term of selections performed per second. The n-gram-based

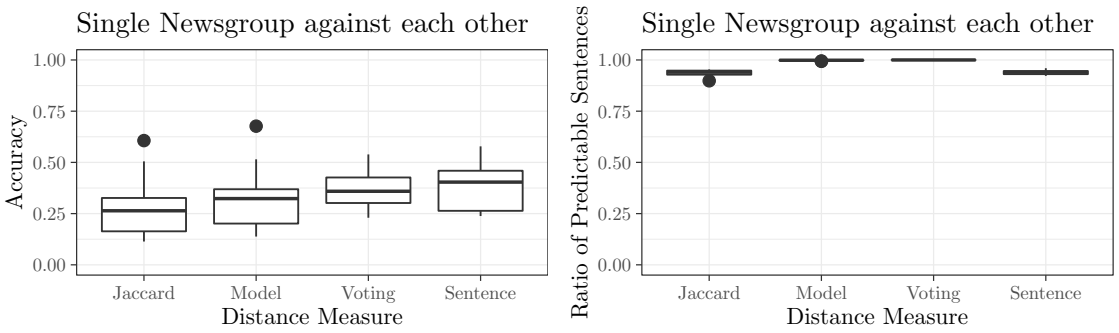


Figure 6.6: Accuracy (left) and ratio of predictable sentences (right) of the Skip-gram-based approach on different newsgroups.

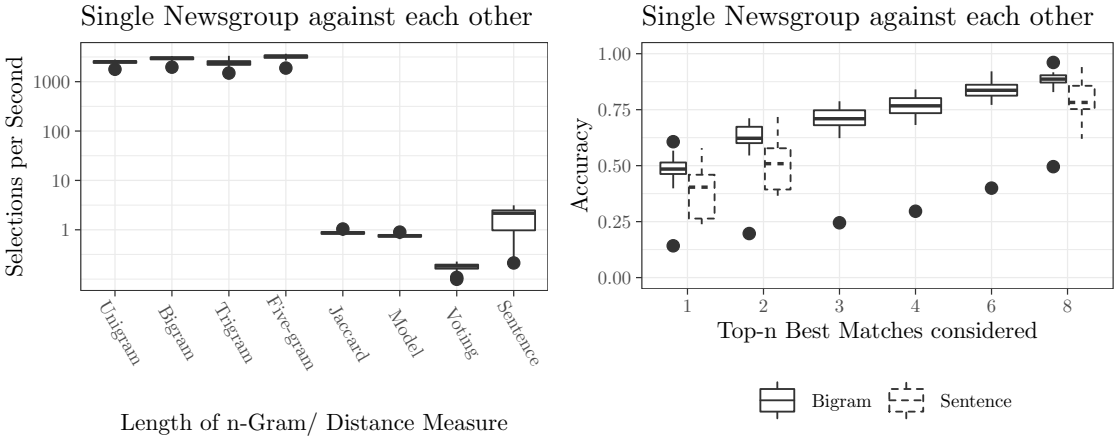


Figure 6.7: Speed (left) of both approaches shown as number of sentences processed per second (logarithmic scaled). Top-n performance (right) using the best parameters of each approach.

approach performs around 3.000 selections per second while the Skip-gram-based approach only performs less than 10 selections in the same time.

Last but not least, we present the performance of the top-n best matching dictionaries on the right side of Fig. 6.7 since an agent might be allowed to return a ranked list of results for a task. We only consider bigrams and the `Sentence` distance measure. An accuracy clearly above 0.5 is already achieved by only considering the top-2 dictionaries. Overall, the n-gram-based approach shows a good performance, especially when using bigrams. In comparison to the Skip-gram-based approach, short run times and good accuracy results are the advantages.

Chapter 7

Adaptation of SCD-word Probability Distributions for other Corpora

Adapting data from one corpus to another corpus is the most common setting in the NLP community and different challenges exist to use data from one corpus in another corpus. Some well-known challenges are given by (i) different representations in both corpora for same entities, (ii) a difference in the context and the vocabulary of documents in both corpora, and (iii) a difference in word-sense distribution, since the sense of a word might depend on the neighboring words. Over the last decades, forms of transfer learning have driven progress for various tasks in NER, ranging from self-supervised learning with auxiliary tasks (Ando and Zhang (2005)), feature alignments (Blitzer *et al.* (2006)), and phrase/word clusters (Lin and Wu (2009)) to deep networks (Glorot *et al.* (2011)), language model embeddings (Peters *et al.* (2017)), and pre-trained language models (Peters *et al.* (2018)). Also other tasks have benefit from transfer learning, e.g., automatic capitalization (Chelba and Acero (2006)), word-sense disambiguation (Komiya *et al.* (2017)), and POS-tagging (März *et al.* (2019)).

Generally, transfer learning techniques involve training a model on data of one domain and apply the model to data in another domain. In the last decade, interest in unsupervised domain adaptation has increased. Unsupervised domain adaptation is the task of modifying a model trained on labeled data available in a source domain to obtain better performance on data available in a target domain, without having labeled data in the target domain.

Unsupervised Domain Adaption deals with the domain shift problem. Generally, labeled data (called source domain) and unlabeled data (called target domain). Both domains contain similar but different data distributions and the objective is to correctly infer the labels on the latter. In this chapter an agent is working with two corpora, denoted as $\mathcal{D}_s$ and $\mathcal{D}_t$, where $\mathcal{D}_s$ refers to the source corpus containing documents associated with located SCDs and $\mathcal{D}_t$ refers to the target corpus containing documents not associated with any SCD. An agent may generate an SCD-word probability distribution $\delta(\mathcal{D}_s)$ from documents in $\mathcal{D}_s$ but cannot generate an SCD-word probability distribution $\delta(\mathcal{D}_t)$ from documents in $\mathcal{D}_t$. Thus, we are interested in associating SCDs from documents in $\mathcal{D}_s$ to documents in $\mathcal{D}_t$, since manually generating new SCDs for documents in

$\mathcal{D}_t$ is a time-consuming and thus expensive task.

Our goal is to use the SCD-word probability distribution $\delta(\mathcal{D}_s)$ to predict the most probably suited SCDs associated with documents in the source corpus and associate them with documents in the target corpus. In our setting, SCDs generate words leading to a generative model s.t. we can estimate the full joint probability distribution of words and SCDs for $\mathcal{D}_s$. Only if the full joint distribution of words and SCDs, represented by $\delta'(\mathcal{D}_s)$, is the same for the source corpus and the target corpus, an agent can use $\delta(\mathcal{D}_s)$ directly to select the MPSCDs and associate them with documents in $\mathcal{D}_t$ (see Chapter 3). However, the full joint distribution of words and SCDs for two different corpora is no the same and it follows that an agent cannot directly use $\delta(\mathcal{D}_s)$ to select MPSCDs for documents in target corpus $\mathcal{D}_t$. Thus, we aim at an adaptation of the source corpus specific full joint probability distribution of $\mathcal{D}_s$ to a target corpus $\mathcal{D}_t$ by looking at the following two different characteristics: (i) word frequency in both corpora, and (ii) topic-word probability distribution of both corpora. Having adopted $\delta(\mathcal{D}_s)$ for $\mathcal{D}_t$, represented as $\hat{\delta}(\mathcal{D}_s)$, an agent can use SCD-word probability distribution $\hat{\delta}(\mathcal{D}_s)$ to automatically enrich documents in target corpus $\mathcal{D}_t$ with SCDs associated with documents in source corpus $\mathcal{D}_s$. Adapting the SCD-word probability distribution for a corpus containing documents associated without any SCD is an important task, since manually generating SCDs for those documents is a time-consuming task.

First, in Section 7.1 we define the problem for adapting SCD-word probability distribution $\delta'(\mathcal{D}_s)$ to documents in a target corpus $\mathcal{D}_t$. Second, in Section 7.2 and Section 7.3, we present two approaches for solving the adaptation problem leading to a SCD-word probability distribution for the target corpus represented as $\hat{\delta}(\mathcal{D}_s)$. Afterwards, we present a case study in Section 7.4 illustrating the effectiveness of both adaptation techniques.

7.1 Domain Adaptation Problem

A corpus $\mathcal{D}_s$ containing documents associated with SCDs represents a supervised learning setting. Generally in a supervised learning setting a set of labeled instances $\{(x_i, y_i)\}_{i=1}^{N}$, $x_i \in \mathcal{X}$, $y_i \in \mathcal{Y}$, where $\mathcal{X}$ represent data and $\mathcal{Y}$ represent labels. Each labeled instance (x_i, y_i) is drawn from a joint probability distribution $p(x, y)$. The joint distribution $p(x, y)$ can be represented by Bayes' rule as follows:

$$p(x, y) = p(x|y)p(y) = p(y|x)p(x) \tag{7.1}$$

In other words, the joint distribution $p(x, y)$ depends on different distributions. In our setting, $p(x|y)$ represents the SCD-word probability distribution $\delta(\mathcal{D}_s)$ generated from all words of documents in $\mathcal{D}_s$ and all SCDs associated with documents available in corpus $\mathcal{D}_s$. Probability distribution $p(y)$ represents the prior probability distribution $p(T(\mathcal{D}_s))$ of all SCDs associated with documents in $\mathcal{D}_s$, where $T(\mathcal{D}_s)$ represents the set of all SCDs

associated with documents in source corpus $\mathcal{D}_s$. We assume the prior probability distribution $p(T(\mathcal{D}_s))$ of SCDs to be uniformly distributed, since a corpus represents a specific context where each SCD might have the same probability of occurrence. We use $p(x)$ to represent the prior probability distribution $p(V(\mathcal{D}_s))$ on vocabulary $V(\mathcal{D}_s)$. The more often a word from vocabulary $V(\mathcal{D}_s)$ appears in documents in corpus $\mathcal{D}_s$, the higher the prior probability of the word. Probability $p(y|x)$ represent the conditional probability of an SCD given words from vocabulary $V(\mathcal{D}_s)$. Since both, words and SCDs are observable in the source corpus we can calculate the source corpus specific joint probability distribution $\delta'(\mathcal{D}_s) = p(V(\mathcal{D}_s), T(\mathcal{D}_s))$ by estimating $\delta(\mathcal{D}_s)$ using Expression (3.2) and multiplying the values in $\delta(\mathcal{D}_s)$ with $p(T(\mathcal{D}_s))$. Following Expression (7.1) we can also represent $\delta'(\mathcal{D}_s)$ in the following way:

$$p(T(\mathcal{D}_s)|V(\mathcal{D}_s))p(V(\mathcal{D}_s)).$$

Generally, if the target corpus $\mathcal{D}_t$ contains the same joint probability distribution as source corpus $\mathcal{D}_s$, we can directly use $\delta(\mathcal{D}_s)$ to automatically enrich documents in $\mathcal{D}_t$ with SCDs associated with documents in corpus $\mathcal{D}_s$. However, the joint probability distribution $\delta'(\mathcal{D}_t)$ of an unlabeled target corpus $\mathcal{D}_t$ is hidden and might differing from the joint probability distribution of the source corpus, since probability distribution $p(V(\mathcal{D}_t))$ differs from probability distribution $p(V(\mathcal{D}_s))$, or hidden probability distribution $\delta(\mathcal{D}_t)$ differs from probability distribution $\delta(\mathcal{D}_s)$.

In both cases, we cannot directly use SCD-word probability distribution $\delta(\mathcal{D}_s)$ to enrich documents in target corpus $\mathcal{D}_t$ with SCDs associated with documents in $\mathcal{D}_s$ and we need to adapt $\delta'(\mathcal{D}_s)$ to $\delta'(\mathcal{D}_t)$ based on the documents in the target corpus. We define the following two cases for adapting SCD-word probability distribution $\delta(\mathcal{D}_s)$ to an unlabeled target corpus $\mathcal{D}_t$.

Case 7.1.1 (Instance Adaptation)**.** An agent cannot directly use the SCD-word probability distribution $\mathcal{D}_s$ to associate an initial set of SCDs with documents in the target corpus $\mathcal{D}_t$ since target probability distribution $p(V(\mathcal{D}_t))$ deviates from source probability distribution $p(V(\mathcal{D}_s))$.

Let us assume that source corpus $\mathcal{D}_s$ and target corpus $\mathcal{D}_t$ contain documents about the same subject, e.g., the Corona virus. However, $\mathcal{D}_s$ contains academic articles and $\mathcal{D}_t$ contains newspaper articles. Then, both corpora contain documents about the same subject but the content is represented in a different style, since the vocabulary of authors writing academic article is different from the vocabulary of authors writing newspaper articles. One reason for the lexical gap between different kinds of documents is given by the difference in both communities of readers. Obviously, an agent cannot use the SCD-word probability distribution $\delta(\mathcal{D}_s)$ for documents in $\mathcal{D}_t$ given a lexical gap between both corpora.

Next, we consider the second case, denoted as labeling adaptation.

Case 7.1.2 (Labeling Adaptation)**.** An agent cannot directly use the SCD-word probability distribution $\mathcal{D}_s$ to associate an initial set of SCDs with documents in the target corpus $\mathcal{D}_t$ since target probability distribution $\delta(\mathcal{D}_t)$ deviates from source probability distribution $\delta(\mathcal{D}_s)$.

Let us assume that source corpus $\mathcal{D}_s$ and target corpus $\mathcal{D}_t$ contain documents about different subjects, e.g., the source corpus $\mathcal{D}_s$ contains documents about the Corona virus and the target corpus $\mathcal{D}_t$ contains old manuscripts from the 1st century. Then, the content of documents from both corpora are contextually different and an agent cannot use the SCD-word probability distribution $\delta(\mathcal{D}_s)$ for documents in $\mathcal{D}_t$.

In Section 7.2 and Section 7.3 we present domain adaptation techniques for the instance adaptation problem and labeling adaptation problem, respectively.

7.2 Instance Adaptation Approach

In case of instance adaptation, the word probability distribution $p(V(\mathcal{D}_t))$ is different from the word probability distribution $p(V(\mathcal{D}_s))$. Generally, an agent can approximate $p(V(\mathcal{D}_s))$ and $p(V(\mathcal{D}_t))$. Afterwards, the agent can adapt the SCD-word probability distribution $\delta(\mathcal{D}_s)$ based on the difference between $p(V(\mathcal{D}_s))$ and $p(V(\mathcal{D}_t))$ resulting in an adapted version of $\delta(\mathcal{D}_s)$ *optimized* for documents in target corpus $\mathcal{D}_t$. The adapted version of $\delta(\mathcal{D}_s)$ is denoted by $\delta(\mathcal{D}_t)$.

The instance adaptation approach focuses on the adaptation of the probability value of each word in $\delta(\mathcal{D}_s)$ based on the difference in word frequencies between both corpora. Algorithm 9 describes the adaptation approach in detail. For both corpora Alg. 9 calculates the corresponding word frequency vectors f_s and f_t to estimate the word probability distributions $p(V(\mathcal{D}_s))$ and $p(V(\mathcal{D}_t))$ for $\mathcal{D}_s$ and $\mathcal{D}_t$, respectively. Afterwards, Alg. 9 estimates the difference between both word probability distributions to obtain an instance adaptation vector v (lines 6-7). Next, Alg. 9 adapts the entries of each row of $\delta(\mathcal{D}_s)$ by multiplying each entry with the corresponding entry in v (*lines 9-16*), resulting in an adapted probability distribution of $\delta(\mathcal{D}_s)$ used as $\delta(\mathcal{D}_t)$.

Proposition 7.2.1. *Algorithm 9 adopts the SCD-word probability distribution of a source corpus optimized for the target corpus $\mathcal{D}_t$.*

Since the domain adaptation problem in Case 7.1.1 lies in the difference between word probability distributions and the words of both corpora are available, we calculate the word probability distributions from both corpora and adapt $\delta(\mathcal{D}_s)$ according to the difference between the two corpus-specific probability distributions (see *line 8*), leading to an SCD-word probability distribution $\delta(\mathcal{D}_t)$ for the target corpus, which solves the problem of Case 7.1.1.

Algorithm 9 Instance Adaptation by Instance Weighting

function InstanceWeighting($\mathcal{D}_s$, $\mathcal{D}_t$)
  Input: Source corpus $\mathcal{D}_s$, target corpus $\mathcal{D}_t$
  Output: adapted SCD-word probability distribution matrix $\hat{\delta}(\mathcal{D}_s)$
  $v \leftarrow$ new zero-vector of length $|V(\mathcal{D}_s)|$
  $\hat{\delta}(\mathcal{D}_s) \leftarrow \delta(\mathcal{D}_s)$
  $f_s \leftarrow$ countFreq($\mathcal{D}_s$)
  $f_t \leftarrow$ countFreq($\mathcal{D}_t$)
  for each $w \in V_{\mathcal{D}_s}$ **do** ▷ Calculate weights
    if $w \in V(\mathcal{D}_t)$ **then**
      $v[w] \leftarrow f_s[w] \cdot (1 - (f_s[w] - f_t[w])) \cdot WF$
  for each row t in $\hat{\delta}(\mathcal{D}_s)$ **do** ▷ Reweight $\delta(\mathcal{D}_t)$
    $c \leftarrow 0$
    for each column w in $\hat{\delta}(\mathcal{D}_s)$ **do**
      $\hat{\delta}(\mathcal{D}_s)[t][w] \leftarrow \hat{\delta}(\mathcal{D}_s)[t][w] \cdot v[w]$
      $c \leftarrow c + \hat{\delta}(\mathcal{D}_s)[t][w]$
    $\hat{\delta}(\mathcal{D}_s)[t] \leftarrow \frac{1}{c} \cdot \hat{\delta}(\mathcal{D}_s)[t]$ ▷ Normalize
  return $\hat{\delta}(\mathcal{D}_s)$
function countFreq($\mathcal{D}$)
  $f \leftarrow$ new zero-vector of length $|V(\mathcal{D}_s) \cup V(\mathcal{D}_t)|$
  $c \leftarrow 0$
  for each $d \in \mathcal{D}$ **do**
    $c \leftarrow c + \#words(d)$
    for each $w \in d$ **do**
      $f[w] \leftarrow f[w] + 1$
  return $\frac{1}{c} \cdot f$ ▷ Normalize

7.3 Labeling Adaptation Approach

As described before, the joint probability distribution $\delta'(\mathcal{D}_s)$ is based on the word distribution $p(V(\mathcal{D}_s))$ and the SCD-word probability distribution $\delta(\mathcal{D}_s)$. Even if documents in both corpora share the same word probability distribution, the SCD-word probability distribution of $\mathcal{D}_s$ and $\mathcal{D}_t$ might be different. Obviously, in this case an agent cannot directly apply $\delta(\mathcal{D}_s)$ for documents in $\mathcal{D}_t$.

In Case 7.1.2, we have asked for an adaptation of the SCD-word probability distribution $\delta(\mathcal{D}_s)$ s.t. an agent can apply $\delta(\mathcal{D}_s)$ for documents in $\mathcal{D}_t$. To estimate where and why $\delta(\mathcal{D}_t)$ differs from $\delta(\mathcal{D}_s)$, we would need some prior about the data in the target corpus. However, the only available data in $\mathcal{D}_t$ are the words in the documents themselves, since no SCDs are associated with documents in $\mathcal{D}_t$. Comparing documents from the source corpus with documents from the target corpus requires a common ground between doc-

Algorithm 10 Labeling Adaptation by Instance Pruning

1: **function** InstancePruning($\mathcal{D}_s, \mathcal{D}_t$)
2: **Input**: Source corpus $\mathcal{D}_s$, target corpus $\mathcal{D}_t$
3: **Output**: adapted SCD-word probability distribution matrix $\delta(\mathcal{C}_{D_t})$
4: Generate topic models $M(\mathcal{D}_s)$, $M(\mathcal{D}_t)$ with $\theta_{\mathcal{D}_s}$, $\theta_{\mathcal{D}_t}$
5: Identify topic mapping σ between $M(\mathcal{D}_s)$ and $M(\mathcal{D}_t)$
6: $\mathcal{C}_{D_t} \leftarrow \emptyset$
7: **for** each $d_t \in \mathcal{D}_t$ **do**
8: **for** each $d_s \in \mathcal{D}_s$ **do**
9: **if** $H_\sigma(\theta_{d_t}, \theta_{d_s}) < \tau$ **then**
10: $\mathcal{C}_{D_t} \leftarrow \mathcal{C}_{D_t} \cup d_s$
11: Build $\delta(\mathcal{C}_{D_t})$ ▷ See Alg. 1
12: **return** $\delta(\mathcal{C}_{D_t})$

uments in both corpora. Generally, we can use any word-based document representation to compare documents from both corpora with each other. A frequently used text-mining technique to represent documents is given by *topic modeling* techniques. Thus, we generate for both corpora $\mathcal{D}_s$ and $\mathcal{D}_t$ a topic model using the well-known latent Dirichlet allocation (LDA) approach and refer to the topic models for $\mathcal{D}_s$ and $\mathcal{D}_t$ by $M(\mathcal{D}_s)$ and $M(\mathcal{D}_t)$, respectively. With a topic model $M(\mathcal{D})$, we can represent and compare two documents $d_i \in \mathcal{D}$ and $d_j \in \mathcal{D}$ by their individual document-topic probability distribution θ_{d_i} and θ_{d_j}, e.g., calculating the Hellinger distance between both probability distributions over K topics and is already defined in Expression (2.12). Based on both corpus-specific topic models $M(\mathcal{D}_s)$ and $M(\mathcal{D}_t)$, we can determine documents in $\mathcal{D}_s$ having a different document-topic probability distribution to documents in $\mathcal{D}_t$ and remove those documents from $\mathcal{D}_s$, resulting in a pruned source corpus $\mathcal{D}_{s'}$. For $\mathcal{D}_{s'}$, we can generate an SCD-word probability distribution $\delta(\mathcal{D}_{s'})$ that is optimized for documents in target corpus $\mathcal{D}_t$, since $\delta(\mathcal{D}_{s'})$ is only based on the documents being topically related to the documents in the target corpus.

Algorithm 10 describes the document pruning approach in detail. Since each corpus represents a specific context, Alg. 10 generates for the source corpus and the target corpus a topic model $M(\mathcal{D}_s)$ and $M(\mathcal{D}_t)$, respectively (*line 4*). Generally, we cannot directly compare the document-topic probability distribution from a document of corpus $\mathcal{D}_s$ with the document-topic probability distribution from a document of $\mathcal{D}_t$ because both document-topic probability distributions are generated from different corpus-specific models s.t. the first topic of $M(\mathcal{D}_s)$ is not the first topic in $M(\mathcal{D}_t)$. Thus, we need a mapping σ between the topics generated from $M(\mathcal{D}_s)$ and $M(\mathcal{D}_t)$, e.g., by analyzing the topic coherence, which is referenced in line 5. We give a detailed analysis about topic mappings in Chapter 9. Next, Alg. 10 generates a cluster $\mathcal{C}_{D_t}$ containing only those documents from source corpus $\mathcal{D}_s$ having a document-topic probability distribution similar

to the document-topic probability distribution of documents in the target corpus, i.e. the similarity is higher than a given threshold τ. Algorithm 10 determines the similarity between two documents based on the Hellinger distance of their document-topic probability distributions. To calculate the Hellinger distance between the document-topic probability distributions of two documents, we need a mapping σ to align the topics of both probability distributions. If the distance between document-topic probability distribution θ_{d_s} and θ_{d_t} is below τ, we add $d_s \in \mathcal{D}_s$ to $\mathcal{C}_{D_t}$ (*line 7-10*), representing a sub-corpus containing documents relevant for the target corpus.

Afterwards, Alg. 10 generates an SCD-word probability distribution $\delta(\mathcal{C}_{\mathcal{D}_t})$ for $\mathcal{C}_{\mathcal{D}_t}$ s.t. $\delta(\mathcal{C}_{\mathcal{D}_t})$ is optimized for the target corpus $\mathcal{D}_t$.

Proposition 7.3.1. *Algorithm 10 adopts the SCD-word probability distribution of a source corpus for a target corpus optimized for the target corpus $\mathcal{D}_t$.*

In Case 7.1.2, we have described that the source probability distribution $\delta(\mathcal{D}_s)$ deviates from the target probability distribution $\delta(\mathcal{D}_t)$. Reasons for $\delta(\mathcal{D}_s)$ not working for $\mathcal{D}_t$ lie in $\delta(\mathcal{D}_s)$ referencing SCDs not relevant for $\mathcal{D}_t$ or SCDs being associated with different words in the context of $\mathcal{D}_t$. Working only with a subset of documents in $\mathcal{D}_s$ has two possible effects, when comparing $\delta(\mathcal{D}_{s'})$ with $\delta(\mathcal{D}_s)$: (i) Pruned documents contain SCDs that are also associated with documents in the reduced corpus $\mathcal{D}_{s'}$, resulting in the same set of SCDs in SCD-word probability distribution $\delta(\mathcal{D}_{s'})$, but different word vector entries. (ii) Pruned documents contain SCDs not associated with other documents resulting in a reduced set of SCDs in $\delta(\mathcal{D}_{s'})$. The effects counteract the two main reasons for $\delta(\mathcal{D}_s)$ not applying to $\delta(\mathcal{D}_t)$. As such, instead of using $\delta(\mathcal{D}_s)$ for $\delta(\mathcal{D}_t)$, we use $\delta(\mathcal{D}_{s'}) = \delta(\mathcal{C}_{D_t})$.

Next, we present a case study illustrating the effectiveness of both adaptation approaches for a source corpus containing documents associated with SCDs and a target corpus containing only documents not associated with any SCD.

7.4 Case Study

After having introduced two unsupervised domain adaptation techniques to adapt the SCDs-word probability distribution $\mathcal{D}_s$ for a target corpus, we present a case study illustrating the performance of both adaptation techniques. In detail, we analyze the performance of both adaptation techniques by calculating the MPSCDs for documents in the target corpus using Alg. 2 before and after adapting the SCD-word probability distribution $\delta(\mathcal{D}_s)$ of a source corpus $\mathcal{D}_s$ for documents in $\mathcal{D}_t$. Next, we describe the data sets, necessary preprocessing techniques, and the evaluation workflow. Finally, we present the results of the domain adaptation techniques.

7.4.1 Data Sets

We have selected articles out of the open and widely accessible online encyclopedia *Wikipedia* to make our experiments reproducible. The data sets contain two sets of articles, which have been grouped by Wikipedia, representing the specific context of a corpus. The SCDs associated with documents in one set represent possibly content descriptions for documents in the other set, since the subjects of both corpora are related.

In the first data set, we use documents about presidents of the United States of America between 1789 and 2017[1] and documents about prime ministers of the United Kingdom between 1721 and 2019[2] as source and target corpus, respectively. Both corpora contain documents about important political persons. In the second data set, we use documents about cities in the United States of America[3] and cities in Europe[4]. Both corpora contain documents about large cities. The number of documents in the source and target corpus is similar for both data sets.

Generally, there are no SCDs associated with Wikipedia articles, because Wikipedia is an online encyclopedia trying to provide objective reference work instead of being a personal reference library containing documents about a specific context. Thus, we have to associate SCDs to documents from the source and target corpus to evaluate the performance of the introduced unsupervised domain adaptation techniques. As stated earlier, SCDs add additional data to documents, making the content explicitly by providing descriptions, references, or explanations about the content. So, we extract data from the text of the Wikipedia articles using Stanford OpenIE (Angeli *et al.* (2015)). OpenIE tools extract relation tuples (RDF triples) directly from the plain text of an article and associate the tuples to the position in the document they have been extracted from. As such, the relational tuples act as located SCDs associated with the documents in this case study. Generally, it is not relevant whether an SCD is manually or automatically associated to a document to evaluate the performance of both domain adaptation techniques. We use automatically generated SCDs to ignore the influence of a single annotator and focus on the adaptation techniques.

7.4.2 Evaluation Setup

Given each data set, we choose one set of articles to be the source corpus $\mathcal{D}_s$ and one set of articles to be the target corpus $\mathcal{D}_t$. We use the located SCDs associated with documents in $\mathcal{D}_t$ as ground truth to evaluate the performance of domain adaptation. We concentrate on those SCDs associated with documents in $\mathcal{D}_t$, which also appear in $\mathcal{D}_s$, as only those can be correctly associated by using $\delta(\mathcal{D}_s)$ or any adapted version of $\delta(\mathcal{D}_s)$.

[1] US president data set - `https://bit.ly/2Z1v1G9`
[2] UK prime ministers data set - `https://bit.ly/3iKbN2W`
[3] US cities - `https://bit.ly/3jUua5H`
[4] European cities - `https://bit.ly/34WXMsE`

For the evaluation, we then remove all SCDs associated with documents in $\mathcal{D}_t$ and use the adaptation techniques to adapt the SCD-word probability distribution $\delta(\mathcal{D}_s)$ of $\mathcal{D}_s$ to estimate the SCD for documents in $\mathcal{D}_t$ and compare the result with the ground-truth SCDs. Specifically, we evaluate the adaptation performance of Alg. 9 and Alg. 10 by comparing the estimated MPSCDs for documents in the target corpus before and after adapting the SCD-word probability distribution $\delta(\mathcal{D}_s)$ using the PPV.

We consider the following four cases using $\delta(\mathcal{D}_s)$ as well as different adapted versions of $\delta(\mathcal{D}_s)$:

(i) Baseline: Using $\delta(\mathcal{D}_s)$ without any adaptation for documents in $\mathcal{D}_t$.

(ii) Instance adaptation: Using Alg. 9 reweighing the influence values in $\delta(\mathcal{D}_s)$ based on the words in $\mathcal{D}_t$ using the best weighting factors (line 10 in Alg. 9) identified before.

(iii) Labeling adaptation: Using Alg. 10 selecting only document from $\mathcal{D}_s$ having a high topic similarity with documents in $\mathcal{D}_t$ to generate a target corpus optimized SCD-word probability distribution from the source corpus.

(iv) Both: Applying both algorithms in sequence, since both probability shifts can occur simultaneously.

7.4.3 Evaluation Workflow

We download all necessary documents from Wikipedia using a Python script and the Wikipedia API and store the documents in the respective corpus. Afterwards, we preprocess the documents by performing the following tasks: (i) lowercase all characters, (ii) stem the words, (iii) tokenize the result, (iv) eliminate tokens from a stop-word list, and (v) extract relation tuples using OpenIE. The first four tasks are standard preprocessing tasks in the NLP community, transforming the text of documents into more digestible form for machine learning algorithms, to increase their performance.

For each data set, the preprocessing steps result in a source and target corpus containing documents that are associated with located SCDs. Then, we evaluate the adaptation performance of both unsupervised domain adaptation techniques by performing the following tasks for each data set:

(i) Identify the SCDs occurring in both corpora to determine the set of SCDs we can correctly associate to documents in target corpus $\mathcal{D}_t$ using the (adapted) SCD-word probability distribution $\hat{\delta}(\mathcal{D}_s)$ of source corpus $\mathcal{D}_s$.

(ii) Remove all SCDs associated with documents in $\mathcal{D}_t$ s.t. $\mathcal{D}_t$ represents a common reference library, where documents contain only text and no SCDs.

(iii) Calculate $\delta(\mathcal{D}_s)$ using Alg. 1.

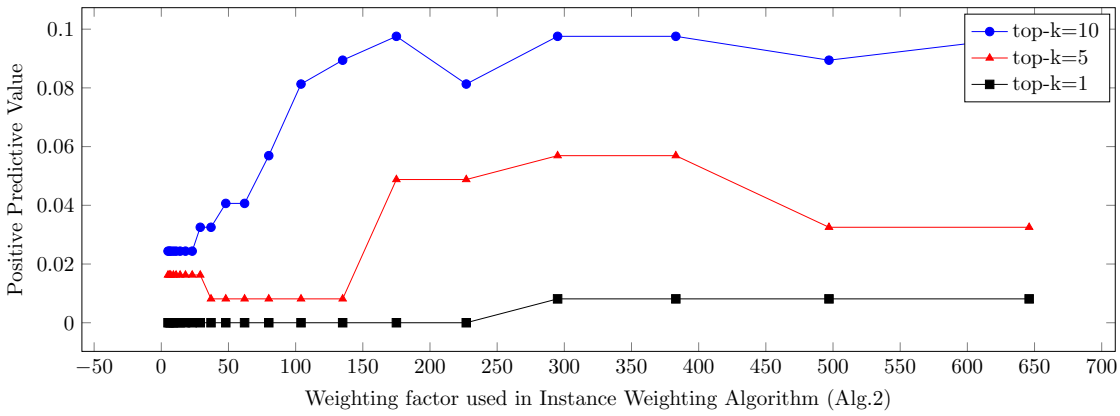


Figure 7.1: Estimating best weighting factors for data set 1 used in Algorithm 9.

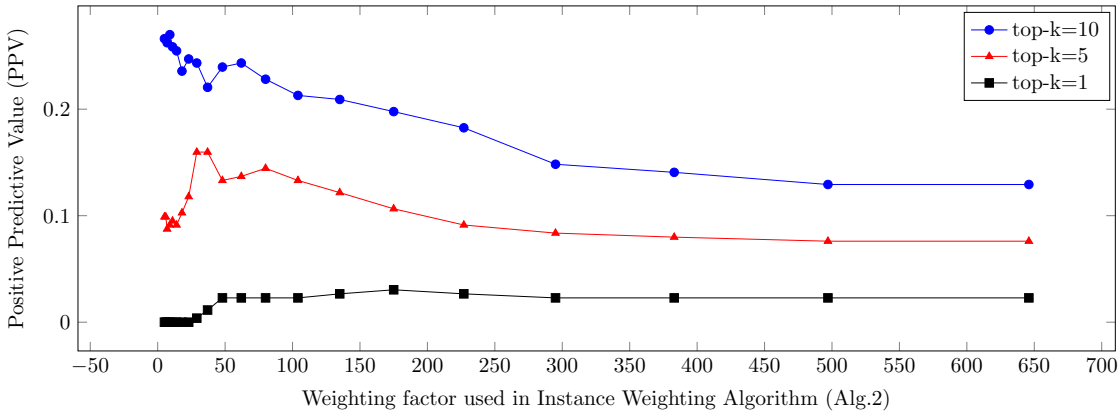


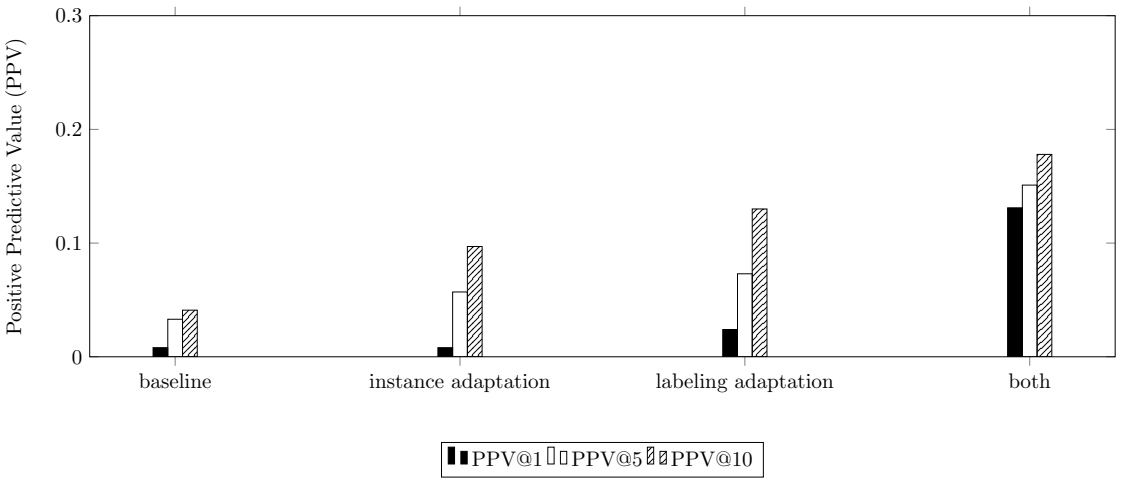
Figure 7.2: Estimating best weighting factors for data set 2 used in Algorithm 9.

(iv) Estimate MPSCDs for documents in $\mathcal{D}_t$ using Alg. 2 with the original SCD-word probability distribution $\delta(\mathcal{D}_s)$.

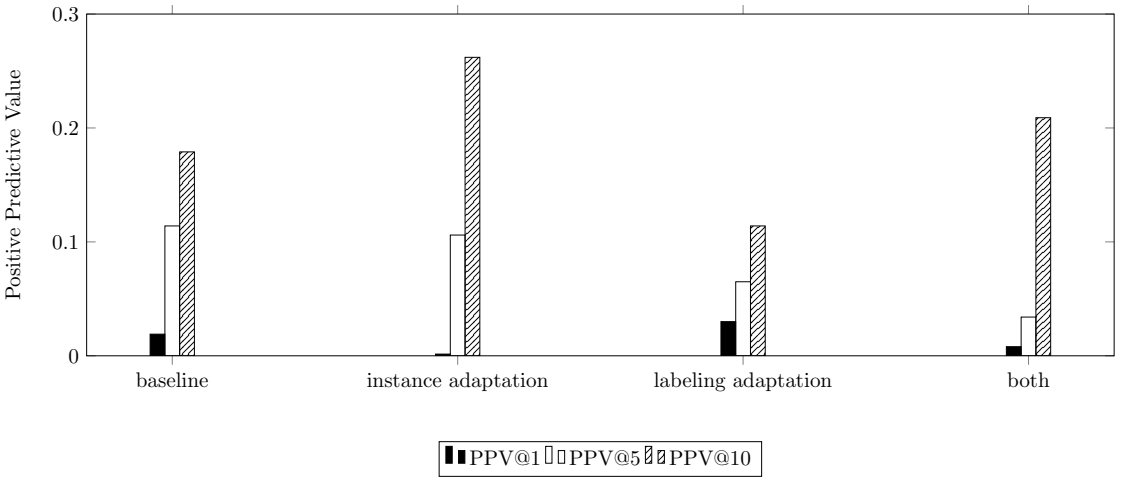
(v) Calculate the baseline PPV for the SCDs of documents in $\mathcal{D}_t$ using the original SCD-word probability distribution $\delta(\mathcal{D}_s)$ s.t. we can compare the performance of both domain adaptation techniques with the baseline PPV.

(vi) Perform instance adaptation (Alg. 9) and labeling adaptation (Alg. 10) on the source corpus and use the adapted versions of $\delta(\mathcal{D}_s)$ to estimate MPSCDs for documents in $\mathcal{D}_t$ using Alg. 2.

(vii) Calculate the PPVs for SCDs associated with documents in $\mathcal{D}_t$ after performing instance and labeling adaptation and compare the performance of both adaptation techniques with the baseline PPV.



(a) Performance of no adaptation, instance adaptation, labeling adaptation, and a combination of both techniques using first labeling followed by instance adaptation (both) for data set 1. We use PPV@1, PPV@5, and PPV@10, to represent the PPV performance looking at the top-k = 1,5,10 MPSCDs, respectively.



(b) Performance of no adaptation, instance adaptation, labeling adaptation, and a combination of both techniques using first labeling followed by instance adaptation (both) for data set 2. We use PPV@1, PPV@5, and PPV@10, to represent the PPV performance looking at the top-k = 1,5,10 MPSCDs, respectively.

Figure 7.3: Adaptation performance for both datasets.

7.4.4 Results

This section presents results regarding the weighting factors as well as the domain adaptation approaches.

Weighting Factor We show the effect of different weighting factors for both data sets in Fig. 7.1 and Fig. 7.2 . For data set 1, higher weighting factors lead to higher PPV. For data set 2, smaller weighting factors lead to higher PPV. As we have expected, the PPV for data set 2 is higher than for data set 1 using only Alg. 9.

Domain Adaption Figure 7.3a and Fig. 7.3b presents the performance of the four cases described previously using the source corpus specific SCD-word probability distribution $\delta(\mathcal{D}_s)$ and three adapted versions of $\delta(\mathcal{D}_s)$.

We evaluate for both data sets the performance of estimating the MPSCDs for documents in the target corpus considering each of the four cases. Algorithm 2 selects the MPSCD based on the similarity value of SCDs. The similarity value of the first k MPSCDs might be almost the same. Thus, we consider the top-k MPSCDs and mark an estimated MPSCD as true positive, if the estimated SCD is in the top-k MPSCDs. We use three different settings considering the top-1, top-5 and top-10 MPSCDs, represented by PPV@1, PPV@5, and PPV@10, respectively.

In case of labeling adaptation, we use 15 topics for the topic model, since the topic model containing 15 topics has the best quality w.r.t the perplexity of the models and use $\tau = 0.6$ (Alg. 10 line 7) as threshold to decide if two documents are similar.

As we have expected, the performance using the original SCD-word probability distribution $\delta(\mathcal{D}_s)$ is low for data set 1, because of the varying context and lexical difference between documents in source and target corpus. Instance adaptation slightly increases the PPV in comparison to the baseline. Pruning documents in the source corpus by Alg. 10 results in even better PPVs. However, the combination of both adaptation techniques leads to best results and the different between PPV@1 and PPV@10 is remarkably small.

For the second data set, the PPV using the original SCD-word probability distribution $\delta(\mathcal{D}_s)$ is clearly higher than for data set 1, since the vocabulary used in source and target corpus for data set 2 is more similar than for data set 1. Optimizing $\delta(\mathcal{D}_s)$ using instance adaptation results in best PPV performance considering the top-10 MPSCDs. Interestingly, the PPV decreases by performing labeling adaptation. One reason for the decreasing performance is given by pruning some documents in the source corpus containing valuable SCDs for documents in the target corpus.

Chapter 8

Interim Conclusion

This part has presented SCDs as a specific kind of subjective annotations for documents in a corpus. The SCDs represent context-specific data about the content of documents in a corpus and the SCDs are associated with locations in documents supporting an agent in corpus-specific document retrieval tasks, e.g., (i) estimating a set of similar documents for a document, (ii) identifying positions of interest within a document, and (iii) classifying new documents into different document categories. The document classification of new documents is based on MPSCDs similarity values.

Compared to available semantic annotation systems, we have focused on document annotation approaches using available and subjective content descriptions associated with documents instead of using externally sources to enrich documents with annotations. Additionally, we have introduced different techniques to enrich new documents with subjective content descriptions associated with other documents from the same corpus. Thus, an agent benefits from the introduced approaches and may increase the performance of its corpus-specific tasks. Furthermore, SCD-based approaches support humans in different tasks, e.g., selecting an appropriate dictionary to identify hidden relationships in documents.

There exist many possible roads for further work on SCDs. In this dissertation, we have focused on approaches based on available SCDs associated with documents in corpora. For new documents containing slightly different content as documents in a given corpus we cannot guarantee that associating available SCDs to the new document describe the content, even if an agent might benefit from those SCDs e.g., for classification task. Thus, another direction of research might be generating new SCDs based on the context represented by a corpus and available SCDs associated with documents. At the end of this part, we have covered the first four contributions of this dissertation, namely:

(1) Context-aware classification of documents using SCDs.

(2) Corpus-driven SCD enrichment of documents.

(3) Identifying SCDs within text by defining the iSCD problem.

(4) Context-aware adaptation of SCDs for documents in a target corpus.

All approaches presented in this part of the dissertation apply a topic model to estimate a similarity value of subjective content descriptions. Generally, changing the collection of documents, e.g., by enriching a corpus with new documents, requires an adaptation of the topic model. The next part of this dissertation contains an approach on maintaining topic models and another approach for optimizing the topic modeling performance using Named-Entity induced links between documents within the topic model.

Part II

Topic Modeling Techniques for Desired Subjective Content Descriptions

Chapter 9

Topic Models for Growing Corpora

In this part, results regarding topic modeling techniques that are relevant for selecting SCDs are presented. This chapter looks at techniques to compare topic probability distributions resulting from different topic models with each other. Additionally, this chapter presents a comprehensive evaluation regarding three different strategies to handle documents extending an individual collection of documents. In many scenarios, an agent is not working with a fixed corpus. From time to time an agent might be faced with new documents and the content of those documents might be relevant for different tasks of an agent, e.g., document retrieval. However, for a new document no corpus-driven document-topic probability distribution exists, because the new document was not part of the corpus while the topic model has been generated from the documents in the corpus.

In Chapter 5, we have presented an approach to enrich documents with SCDs. Unfortunately, for corpus-extending documents containing SCDs, we cannot directly use the iterative SCD enrichment algorithm, since the algorithm is based on the document-topic probability distributions (D-similarity) as well as the similarity of SCDs (T-similarity). An agent faced with a new document needs to estimate the document-topic probability distribution for the new document based on the corpus-driven topic model s.t. the agent can use the D-similarity to compare new documents with other documents in a corpus.

Thus, in this chapter we apply techniques to estimate a corpus-driven topic probability distribution for a new document an agent is faced with and might be interested in to extend its corpus with. Additionally, we apply different know techniques to update the initial topic model of a corpus based on new documents extending the initial corpus.

The following paper presented topic model adaptation for growing corpora:

> Felix Kuhr, Magnus Bender, Tanya Braun, and Ralf Möller. Maintaining topic models for growing corpora. In *IEEE 14th International Conference on Semantic Computing, ICSC 2020, San Diego, CA, USA, February 3-5, 2020*, pages 451–458, 2020

The remainder of this chapter is structured as follows: Section 9.1 gives an overview of the known topic model adaptation techniques for growing corpora and describes the *OnlineLDA* approach of Hoffman *et al.* (2010), OLDA for short, as well as *fold-in Gibbs sampling* being detailly described in Griffiths and Steyvers (2004). Section 9.2 describes

how to estimate the topic probability distribution of a new document and how to compare different topic models with each other, since it is non-trivial to identify for a topic in one model the corresponding topic in another model even if the corpus is the same for both models. Section 9.3 presents an empirical evaluation for different known topic model adaptation techniques considering their runtime and classification performance.

9.1 Adapting Topic Models

Topic models are statistical models for discovering hidden semantic structures in the text of documents that are organized in a corpus. Those hidden semantic structures are also known as *topics*. As described in Section 2.2.1, topic modeling approaches reduce documents to a fixed number of topics s.t. an agent can work with the documents at their topic level, e.g., comparing documents not by the words occurring in the documents but also by their document-topic probability distribution. Over time, an agent is faced with new documents and has to decide whether it should extend its corpus with those additional documents or not. For a new document the agent has no document-topic probability distribution, since a corpus represents an individual collection of documents, and generally no topic model has been generated before for a corpus containing the initial documents of the corpus and the new document. Handling possibly corpus-extending documents results in two important tasks we are interested in, namely: (i) estimating the document-topic probability distribution for possibly corpus-extending documents based on the documents in a given corpus, and (ii) incorporating word distributions from corpus-extending documents into an available topic model for the initial corpus. Incorporating corpus-extending documents into a topic model and estimating the document-topic probability distribution of corpus-extending documents enables an agent to identify a set of similar documents in the corpus to those new documents based on the similarity of document-topic probability distributions. Thus, an agent benefits from a corpus-driven document-topic probability distribution for new documents, since the agent can automatically associated SCDs with documents that are already associated with documents in the corpus as well as associating SCDs with new documents extending a corpus. Generally, the following three strategies are available for estimating a corpus-driven document-topic probability distribution for new documents extending an initial corpus:

From scratch after document extension. Extending an initial corpus with a new document and estimating a new topic model from the extended corpus. This strategy results in document-topic probability distribution for the new document, s.t. an agent can compare all documents in the extended corpus by the estimated document-topic probability distributions. We refer to this strategy by the term *LDA*, since we use LDA as the topic modeling approach to estimate a topic model from documents in a corpus.

Incrementally with model modification. Inferring document-topic probability distribution of a new document based on the available topic-word probability distributions of the topic model generated from documents in the initial corpus. Afterwards, adapting the initial topic model based on the new document extending the corpus. We refer to this strategy by the term *onlineLDA*, since we use the onlineLDA topic modeling approach, introduced by Hoffman *et al.* (2010), to update the initial topic model based on the new document.

Incrementally without model modification. Inferring the document-topic probability distribution of a new document is based on the available topic-word probability distributions of the topic model generated from documents in the initial corpus without adapting the initial model. This inferering technique is based on Gibbs sampling (Geman and Geman (1984)) and better known as fold-in Gibbs sampling, since a new document is added into the initial corpus and afterwards Gibbs sampling is performed only on the new document fixing the *old* parameters of the model, s.t. an agent can compare the document-topic probability distribution of a new document with the document-topic probability distribution of documents from the initial corpus. Additionally, the topic-word probability distribution of the initial model is adopted based on the new documents extending an initial corpus.

The first two strategies incorporate that a topic model should represent all documents in its corpus, including new corpus-extending documents, which is why in (i) a complete new model is learned from scratch after document extension is performed and in (ii) the initial topic-word probability distribution is adapted based on new documents extending the corpus. Generally, adapting a topic model is faster than estimating a complete new model since the existing topic model is not dropped and an agent does not need to compute a new topic model from scratch. The strategy in (iii) leaves the initial topic model unchanged, which accepts inaccuracy of the topic model representing all documents in the corpus in exchange for fast processing of new documents.

All three strategies can handle corpus-extending documents and estimate a document-topic probability distribution for new documents based on the documents in the corpus, s.t. an agent can compare documents with each other using their document-topic probability distribution and finally perform corpus-driven SCD enrichment not only for the initial documents in a corpus but also for new documents an agent is interested in. However, the three strategies differ in their performance for various scenarios, e.g., extending the corpus with a single document, a series of single documents, or a batch of many documents. Additionally, the three strategies differ in their performance by adding documents sharing similar content or documents with somehow unrelated content. We analyze the performance of LDA, onlineLDA, and fold-in Gibbs sampling given varying scenarios an agent might be faced with.

Formally, for each document $d \in \mathcal{D}$, LDA estimates a discrete document-topic probability distribution θ_d over the K topics, which contains for each topic $k \in \{1, \ldots, K\}$ a value between 0 and 1 s.t. the sum of all values is 1, and a discrete probability distribution ϕ_k for each topic $k \in \{1, \ldots, K\}$ over the words in $\mathcal{V}$, which contains for each $w \in \mathcal{V}$ a value between 0 and 1 s.t. the sum of all values is 1. Both probability distributions represent a corpus-driven topic model $\mathcal{M}(\mathcal{D})$. As already presented in Section 2.2.1 we define a topic model by:

$$\mathcal{M}(\mathcal{D}) = (\theta_d, \phi_k), \tag{9.1}$$

where $d \in \mathcal{D}$, and $k \in \{1, \ldots, K\}$. For further details, please refer to Section 2.2.1.

Hoffman *et al.* (2010) have introduced onlineLDA (OLDA) to analyze large collections of documents. OLDA is optimized for handling streams of documents extending a corpus and efficiently adapts topic-word probability distributions to calculate document-topic probability distributions for new documents by approximating the posterior probability in Expression (2.10) using online stochastic optimization converging to a local optimum of a variational Bayes objective function. To extend an initial corpus $\mathcal{D}$ with a new document d', OLDA updates the initial topic-word probability distributions $\phi_k, k \in \{1, \ldots, K\}$ and the document-topic probability distributions $\theta_d, d \in \mathcal{D} \cup \{d'\}$ by using an EM-algorithm iterating over the extended corpus until the model performance converges or a fixed number of iterations is reached. For further details, please refer to Alg. 2 in Hoffman *et al.* (2010). In contrast to using the LDA technique for learning a new model, OLDA reuses the probability distributions of the *old* topic model by adapting the model based on the words within the new, corpus-extending documents.

Generally, adapting a topic model given a new document might drag down the performance of the topic model on the original documents, though. The probability distributions estimated by OLDA and LDA are slightly different. In Section 9.3, we evaluate the difference between both models.

FIGS refers to adding a new document d' to the initial corpus $\mathcal{D}$ and performs Gibbs sampling (Griffiths and Steyvers (2004)) only for d' using the words in d'. FIGS does not adapt the initial document-topic probability distributions $\theta_d, d \in \mathcal{D}$ and the topic-word probability distributions $\phi_k, k \in \{1, \ldots, K\}$. Thus, FIGS is even faster than OLDA, but ignores the content of new documents for the document-topic probability distributions already estimated for documents in the corpus. FIGS assigns the most probable topic for each word $w \in d'$ using the topic-word probability distributions $\phi_k, k \in \{1, \ldots, K\}$. Then, FIGS computes for each word in d' the probability being assigned to each of the K topics, samples a topic from the document's topic probability distribution and assigns the word to the new topic. If d' contains a new word w not part in any document $d \in \mathcal{D}$, the Gibbs sampling process randomly assigns a topic for this word. The topic assignment of the words in d' yields the distribution of topics in d'. FIGS requires only few iterations taking the topic structure into account, i.e., allocate words of documents to as few topics as possible.

Technically, there is no limitation in the number of documents for extending a corpus using OLDA or FIGS, but the document-topic probability distributions $\theta_d, d \in \mathcal{D}$, topic-word probability distributions $\phi_k, k \in \{1, \ldots, K\}$, and the optimal number of topics (K) changes with each corpus-extending document.

Thus, an agent should generate a new topic model after a while.

9.2 Maintaining Topic Models

This section discusses how to compare document-topic probability distributions with each other and how to compare topic models with each other, where topics are not necessarily matching one-to-one between different topic models, i.e. the first topic-word probability distribution of one model and the first topic-word probability distribution of another model does not necessarily represent the same topic.

9.2.1 Comparing Document-Topic Probability Distributions within a Topic Model

We can use the Hellinger distance to compare document-topic probability distributions and topic-word probability distributions for a single topic model $\mathcal{M}(\mathcal{D})$ generated from the documents in a corpus $\mathcal{D}$, since Hellinger distance allows for measuring the distance between two probability distributions. Given document-topic probability distributions θ_{d_i} and θ_{d_j} for two documents $d_i, d_j \in \mathcal{D}$, the Hellinger distance $H(\theta_{d_i}, \theta_{d_j})$ between θ_{d_i} and θ_{d_j} is defined in Expression (2.12). For the readers convenience we present the Hellinger distance at this position, again:

$$H(\theta_{d_i}, \theta_{d_j}) = \frac{1}{\sqrt{2}} \sqrt{\sum_{k=1}^{K} \left(\sqrt{\theta_{d_i,k}} - \sqrt{\theta_{d_j,k}} \right)^2}$$

Since the number of topics K is usually small, it is computationally feasible to calculate the Hellinger distance $H(\theta_{d_i}, \theta_{d_j})$ between two distributions, since we sum over the K topics by using the Hellinger distance. To compute the Hellinger distance between two topic-word probability distributions ϕ_{k_i}, ϕ_{k_j}, the inner sum of Expression (2.12) goes over the words in the vocabulary of $\mathcal{D}$ and assumes that the two distributions are indexed over the *same* topics, i.e., K topics in case of document-topic probability distributions $\theta_{d_i}, \theta_{d_j}$ and vocabulary $\mathcal{V}$ in case of topic-word probability distributions ϕ_{k_i}, ϕ_{k_j}.

Comparing distributions from two different topic models $\mathcal{M}(\mathcal{D})$ and $\mathcal{M}'(\mathcal{D})$ calculated from the same collection of documents ($\mathcal{D}$), the assumption may be violated. In case of topic-word probability distributions ϕ_k and $\phi_{k'}$, it is reasonable to assume the vocabulary is the same since the corpus is the same. Therefore, words in topics can be matched between ϕ_k and $\phi_{k'}$. But, assuming that K is identical for both $\mathcal{M}(\mathcal{D})$ and $\mathcal{M}'(\mathcal{D})$, the

K topics, over which the inner sum iterates, may not be as easily matched. Generally, topics are only abstract structures representing a probability distribution over the words in a vocabulary, i.e., ϕ_k for all $k \in \{1, \ldots, K\}$ and does not contain any name like *sports* or *education*. Thus, estimating the document-topic probability distributions does not guarantee that topic $k = 1$, represented by ϕ_1 in $\mathcal{M}(\mathcal{D})$, matches the topic $k = 1$, represented by ϕ_1 in $\mathcal{M}'(\mathcal{D})$.

Next, we present techniques to match topics from one topic model with topics from another topic model.

9.2.2 Matching Topics from Different Topic Models

As already mentioned, comparing document-topic probability distributions of documents with each other from different topic models is difficult, since topics have no names and the first topic from a model $\mathcal{M}(\mathcal{D})$ does not necessarily represent the first topic from another model $\mathcal{M}'(\mathcal{D})$. Thus, we need a technique mapping K topics from one topic model $\mathcal{M}'(\mathcal{D})$ to K' topics from another model $\mathcal{M}'(\mathcal{D})$ s.t. an agent can compare the topics from different models with each other. We say the best match between the topics of two topic models $\mathcal{M}(\mathcal{D}), \mathcal{M}'(\mathcal{D})$ is given by the mapping σ of the K topics from $\mathcal{M}(\mathcal{D})$ to the K' topics of $\mathcal{M}'(\mathcal{D})$ that has the minimal sum of the Hellinger distances between the topic-word probability distributions.

Mathematically, we describe the identification of the best mapping between the topics of two models as an optimization problem. Expression (9.2) presents the optimization problem as a minimization problem:

$$\arg\min_{\sigma} \sum_{k=1}^{K} H(\phi_k, \phi'_{\sigma(k)}), \tag{9.2}$$

where σ denotes a mapping function that maps each topic $k \in \{1, \ldots, K\}$ in $\mathcal{M}(\mathcal{D})$ to a topic $k' \in \{1, \ldots, K'\}$ in $\mathcal{M}'(\mathcal{D})$.

If $K = K'$, we may impose bijectivity on σ to require that each topic in $\mathcal{M}(\mathcal{D})$ is mapped to exactly one topic in $\mathcal{M}'(\mathcal{D})$, and vice versa. We consider the following three techniques estimating the best mapping between the topics of $\mathcal{M}(\mathcal{D})$ and $\mathcal{M}'(\mathcal{D})$:

(i) **Full Permutation:** Calculate the Hellinger distance for each possible mapping between the topics of $\mathcal{M}(\mathcal{D})$ to the topics of $\mathcal{M}'(\mathcal{D})$ to identify the best mapping, i.e., exactly determine Expression (9.2), yielding a bijective mapping between both topic models. The complexity of full permutation is given by $O(K!T_H)$, where T_H refers to the complexity of calculating the Hellinger distance, which depends on the number of topics K. Generally, this technique is only applicable for very small K.

(ii) **Topic coherence:** Estimate for each topic k in topic model $\mathcal{M}(\mathcal{D})$ and for each topic k' in model $\mathcal{M}'(\mathcal{D})$ those documents having a high document-topic probability distribution for the respective topic (*top-doc*) and compare the topics between

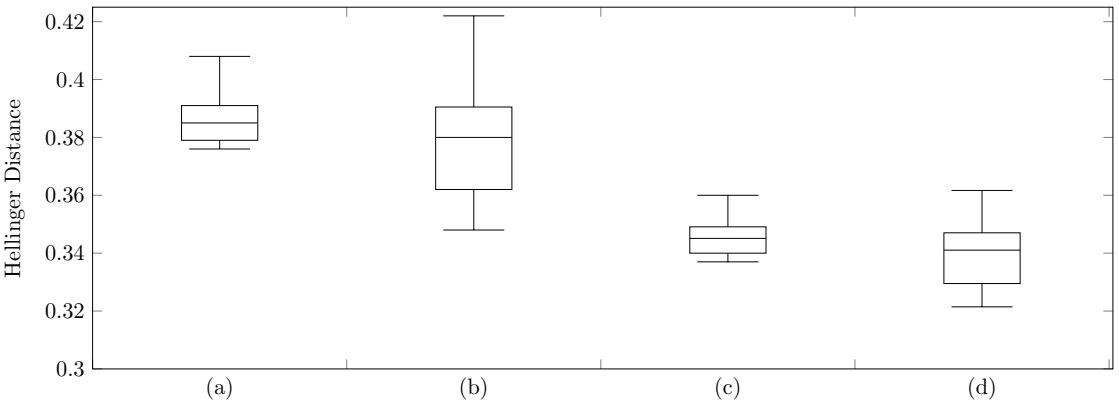


Figure 9.1: Topic mapping approaches of different topic models estimated for the same corpus using (a) a random mapping, (b) topic coherence, (c) best permutation, and (d) minimal Hellinger distance for 50 topic models from the same corpus.

$\mathcal{M}(\mathcal{D})$ and $\mathcal{M}'(\mathcal{D})$ in both directions using the Jaccard coefficient J on the assigned documents. This technique results in two sets, where each set contains for each topic a set of documents. Additionally, we compare the *top-c* words of each topic k in model $\mathcal{M}(\mathcal{D})$ with the *top-c* words of all topics k' in model $\mathcal{M}'(\mathcal{D})$ using the Jaccard coefficient J, resulting again in two sets of document-topic assignments. Thus, for each topic k in model $\mathcal{M}(\mathcal{D})$, we have four possible topic mappings to topics in model $\mathcal{M}'(\mathcal{D})$. We use a majority vote to map k to k'.

The basic assumption for the topic coherence technique is, that documents characterize a topic. The mapping between topics is not necessarily bijective as two or more topics from model $\mathcal{M}(\mathcal{D})$ might be mapped to the same topic in $\mathcal{M}'(\mathcal{D})$ given the topic-assigned documents. The advantage of this technique is its superior runtime in $O(K^2 \cdot T_J)$, where T_J refers to the complexity of calculating J, which depends on the number of documents in the corpus.

(iii) **Minimal Hellinger distance:** Calculate the Hellinger distance between each topic k in $\mathcal{M}(\mathcal{D})$ and all topics k' in $\mathcal{M}'(\mathcal{D})$.

$$k' = \underset{k \in \{1,\dots,K'\}, k' \in \{1,\dots,K'\}}{\arg\min} H(\phi_k, \phi'_{k'})$$

Again, the mapping is not necessarily bijective. Compared to topic coherence, the best match is based not on the *top-doc* documents but on the distribution over all topics. The complexity is $O(K^2 \cdot T_H)$, where T_H again refers to the complexity of calculating the Hellinger distance.

We generate 50 topic models from documents in a corpus $\mathcal{D}$ to identify the best mapping technique, since the inferred document-topic probability distributions of new docu-

ments and the document-topic probability distributions of the same documents generated by a new topic model for $\mathcal{D} \cup \{d'_i\}_{i=0}^{T}$ depend on (i) documents in corpus $\mathcal{D}$, (ii) the length of the documents in $\mathcal{D}$, and (iii) the words in the documents. Generating more than 50 topic models did only slightly change the performance of the comparison. Generally, it is not relevant that we have used exactly 50 topic models, but using more models reduce the randomness.

We compare the three techniques and the average distance between randomly selected mappings for topic-word probability distributions of two topic models $\mathcal{M}(\mathcal{D}), \mathcal{M}'(\mathcal{D})$ learned for one corpus $\mathcal{D}$ (without extending $\mathcal{D}$). Figure 9.1 presents the Hellinger distance of the matched topics given the mapping σ generated by (a) random mapping – randomly selecting one mapping for each of the 50 topic models, (b) topic coherence, (c) best permutation – selecting for each of the 50 topic models only its best mapping, and (d) minimal Hellinger distance.

The average performance of all mapping approaches is similar having a Hellinger distance between 0.3 and 0.4. However, the performance of the topic coherence technique varies the most and the performance of the best mapping from the full permutation technique the least. The best performance can be reached using the minimal Hellinger distance in (iii), which is what we will use later for the evaluation. Since the minimal Hellinger distance allows us to map two topics in $\mathcal{M}(\mathcal{D})$ to the same topic in $\mathcal{M}'(\mathcal{D})$, it is possible that the performance of the minimal Hellinger distance is better than the best results from a full permutation that is a bijective mapping.

Given a way to match topics from different topic models, next we specify two ways to compare topic models with each other.

9.2.3 Comparing Topic Models

We are interested in evaluating the performance of the three strategies LDA, OLDA, and FIGS in estimating document-topic probability distributions of a new document an agent is faced with and possibly might extend its initial corpus $\mathcal{D}$ with.

A famous measure in the NLP community for language models is *perplexity*. The perplexity of a topic model describes how well the generated model predicts a sample. The smaller the perplexity of a topic model, the better is the prediction performance of the model for samples. Besides perplexity, we focus on the document-topic probability distribution of documents as a measure to compare the performance of the different strategies.

Generating two topic models from the same collection of documents in a corpus $\mathcal{D}$ using the same initial topic modeling parameters leads to two different topic models distinguishing in their topic-word probability distribution ϕ and the document-topic probability distribution θ for each document in $\mathcal{D}$, i.e., the document-topic probability distribution θ_d of document d generated by one topic model $\mathcal{M}(\mathcal{D})$ is different from θ_d generated by another model $\mathcal{M}'(\mathcal{D})$ from the same composition of documents in $\mathcal{D}$. One reason for the

difference in the document-topic probability distribution is given by the approximative inference algorithms estimating the topic-word probability distributions and document-topic probability distributions. Thus, we say that there is an "excused error" that we attribute to the approximate nature of the calculations. We call this error a baseline error $b_{err}(\mathcal{M}(\mathcal{D}), \mathcal{M}'(\mathcal{D}))$ between two topic models $\mathcal{M}(\mathcal{D})$ and $\mathcal{M}'(\mathcal{D})$. We define the baseline error by:

$$b_{err}(\mathcal{M}(\mathcal{D}), \mathcal{M}'(\mathcal{D})) = \frac{\sum_{k=1}^{K} H(\phi_k, \phi_{\sigma(k)})}{K},$$

where $H(\phi_k, \phi_{\sigma(k)})$ represents the Hellinger distance between topic-word probability distribution ϕ_k and the corresponding topic-word probability distribution $\phi_{\sigma(k)}$ estimated by the topic mapping. Generally, calculating Hellinger distance H requires a mapping σ between both topic models $\mathcal{M}(\mathcal{D})$, and $\mathcal{M}'(\mathcal{D})$ for the inner sum of Expression (2.12).

Having a baseline error, we can define the classification performance $\mathcal{K}$ for evaluating the performance between LDA, OLDA, and FIGS in estimating the document-topic probability distributions for an extend corpus $\mathcal{D}' = \mathcal{D} \cup \{d'_i\}_{i=0}^{T}$ containing T new documents as follows:

$$\mathcal{K}(\mathcal{M}(\mathcal{D}), \mathcal{M}'(\mathcal{D})) = \max\left(0, \frac{\sum_{d \in \mathcal{D}'} H(\theta_d, \theta'_d)}{|\mathcal{D}'|} - b_{err}(\mathcal{M}(\mathcal{D}), \mathcal{M}'(\mathcal{D}))\right),$$

where $\mathcal{M}'(\mathcal{D})$ represents the topic model generated by LDA and represents the ground-truth for the document-topic probability distributions and topic-word probability distributions. $\mathcal{M}(\mathcal{D})$ represents the topic model generated by FIGS (OLDA) so that we can compare the performance of FIGS (OLDA) with LDA generating a completely new topic model from an extended corpus, while considering the baseline error. $\mathcal{K}$ indicates the average Hellinger distance of the document-topic probability distributions. The smaller the classification value of $\mathcal{K}$, the better is the performance of a model.

9.3 Evaluation of Topic Models for Growing Corpora

This section presents an empirical evaluation of LDA, OLDA, and FIGS in the context of growing corpora, where LDA presents the baseline performance. Each of the three strategies estimates a document-topic probability distribution given the text of corpus-extending documents. However, the performance of the three strategies differs in the (i) perplexity for the overall corpus, (ii) held-out set perplexity, (iii) classification performance, and (iv) runtime for corpora of different size.

9.3.1 Evaluation Scenarios

This evaluation focuses on which strategy performs best for different scenarios, extending the corpus by adding batches of 1000 corpus-extending documents. We compare the

performance of OLDA and FIGS against the performance of LDA, acting as the baseline, on the well-known data set *20Newsgroup* (Lang (1995)). The data set contains 20 different newsgroups, each corresponding to a different topic. However, some of the newsgroups are closely related, e.g., *autos* and *motorcycles*, *baseball* and *hockey*, or *pc hardware* and *mac hardware*, resulting in 11 to 13 distinct topics. We look at the following two types of unseen document:

(Type 1) the categories of unseen documents are not part of the documents in the initial corpus or

(Type 2) the documents in the initial corpus contain the same categories as the corpus-extending documents.

We expect FIGS and OLDA to perform better with documents of a known category compared to documents of an unknown category.

There are different options to add documents to a corpus, e.g, document by document or a set of documents by a set of documents. We are interested in the following two settings for OLDA to understand whether documents should be added incrementally to the corpus or not:

(incr.) Short for incremental, where in each iteration, we add a batch of documents to the corpus, perform OLDA, and continue with the extended corpus when adding the next batch of documents in the next iteration. In this evaluation we add in each iteration a batch of 1.000 documents to the growing corpus.

(init.) Short for initial, where we retain the original corpus and add in each iteration an increasingly larger batch of documents to the corpus. This means in the first iteration, we add 1.000 documents to the corpus. In the second iteration, we add the initial 1.000 documents plus additional 1.000 documents to the original corpus, and so on. We increase the size of the batches to analyze which size of a batch results in which performance.

We are interested in the performance for the documents in a corpus and the held-out set for the *incremental* and *initial* techniques extending the corpus, since an agent is faced with new from time to time and not only once.

Preprocessing We preprocess 18.846 documents in the corpus by (i) lowercasing all characters, (ii) stemming the words, (iii) tokenizing the result, and (iv) eliminating tokens part of a stop-word list.

9.3.2 Perplexity

Perplexity is a measurement to estimate how well a probability distribution predicts a sample. The smaller the perplexity the better a probability distribution predicts a

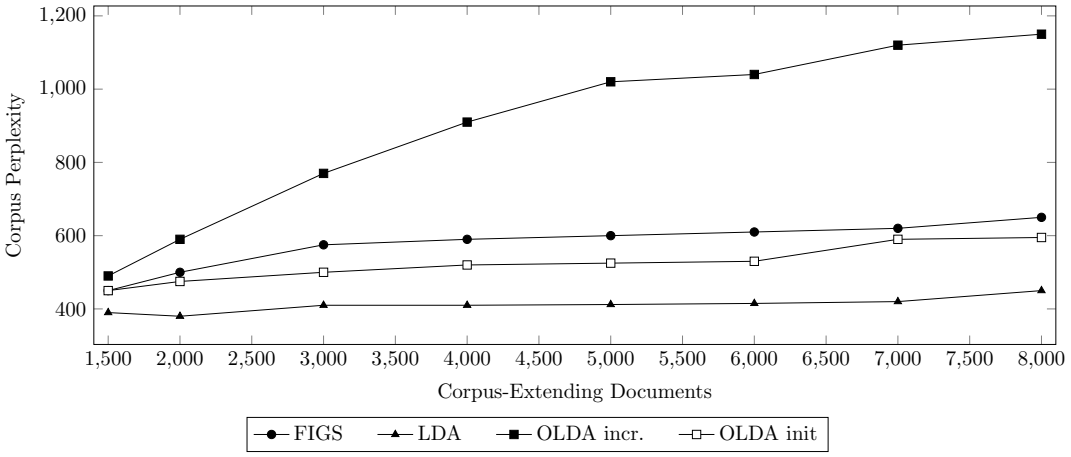


Figure 9.2: Perplexity using LDA, FIGS, OLDA incr., and OLDA init. We present corpus-extending strategy using *Type 1*. Setting: initial corpus size: 8k, α: 0.1, β: 0.1, topics: 11, iterations: 10k.

sample. Mathematically, the perplexity is defined as follows:

$$perplexity(w) = \exp\left(-\frac{\log\left(p(w)\right)}{\sum_{d=1}^{D}\sum_{j=1}^{V} n^{(jd)}}\right),$$

where $n^{(jd)}$ represents the occurrence of the j-th word in document d. The smaller the perplexity, the better the quality of the model.

We present the topic model perplexity of LDA, OLDA, and FIGS for the following two cases. In the first case (Type 1), the categories of unseen documents are not part of the documents in the initial corpus. In the second case (Type 2), the documents in the initial corpus contain the same categories as the corpus-extending documents. For LDA, a new topic model is calculated from the initial collection of documents in corpus $\mathcal{D}$ and the set of corpus-extending documents $\{d'_i\}_{i=0}^{T}$. For FIGS, we use the available topic-word probability distributions ϕ_k, $k \in \{1, ..., K\}$ and document-topic probability distributions θ_d, $d \in \{1, ..., |D|\}$ of the initial corpus $\mathcal{D}$ for inferring the document-topic probability distribution $\theta_{d'_i}$ for each corpus-extending document in $\{d'_i\}_{i=0}^{T}$. In the setting of OLDA, we update the initial topic model after extending the corpus with T corpus-extending documents $\{d'_i\}_{i=0}^{T}$. The performance of OLDA depends on the categories occurring in the documents of the initial corpus and the categories of the corpus-extending documents. Generally, the performance of OLDA is worse when the categories of unseen documents are not part of the documents in the initial corpus.

In Figure 9.2, we evaluate the topic model perplexity of FIGS, LDA, and both variants

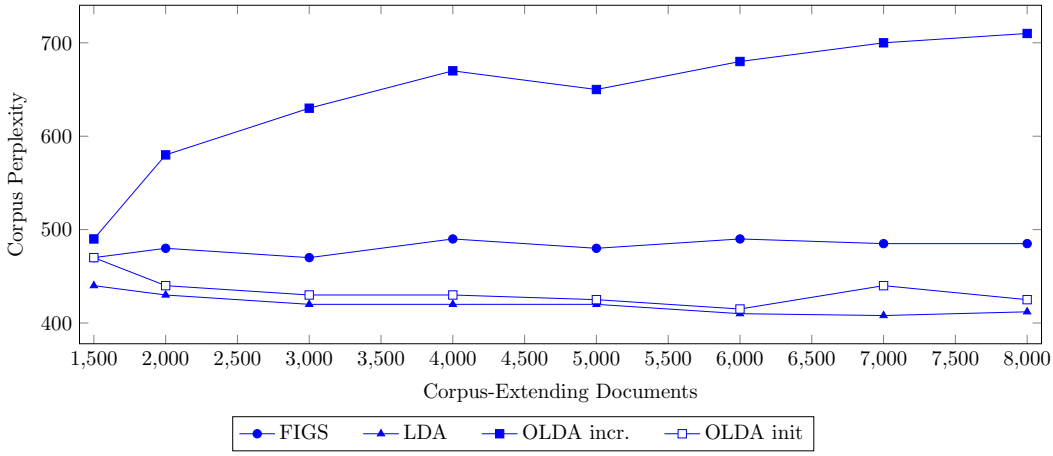


Figure 9.3: Perplexity using LDA, FIGS, OLDA incr., and OLDA init..We present corpus-extending strategy using *Type 2*. Setting: initial corpus size: 8k, α: 0.1, β: 0.1, topics: 11, iterations: 10k.

of OLDA considering only unseen documents of Type 1. The corpus perplexity increases with corpus-extending documents for the incremental variant of OLDA. Adding all batches to the initial corpus and adapting the initial topic model using OLDA init leads to a similar corpus perplexity as using LDA which calculates a new topic model from all documents. Generally, the perplexity is lower if the initial corpus contains documents that share the same newsgroup as the corpus-extending documents (Type 2). Figure 9.3 presents the evaluation of the topic model perplexity for FIGS, LDA, and both variants of OLDA considering only unseen documents containing the same categories as the documents in the initial corpus. Again, the performance of OLDA incr is worst. For all three other techniques, the corpus perplexity is similar.

In Figure 9.4, we present the perplexity of a fixed hold-out set of documents using FIGS, LDA, and OLDA for Type 1 (left) and Type 2 (right). Initially, we generate one topic model from all documents within a corpus $\mathcal{D}$ containing 10k documents. In each step, we extend $\mathcal{D}$ with a batch of 1k documents and calculate the perplexity for the new documents by performing the following steps:

(i) For LDA, adding documents to the initial corpus and calculate a new topic model from the extended corpus,

(ii) for OLDA incr., in each step adding the new documents to the actual corpus and update the actual topic model resulting in a new document-topic probability distribution for all documents in the corpus, and

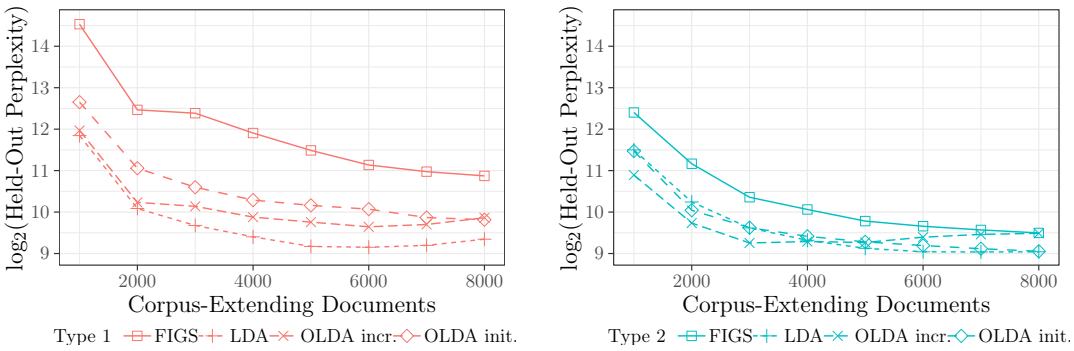


Figure 9.4: Both plots show the perplexity using LDA, FIGS and both variants of OLDA for a held-out set.

(iii) for OLDA init., adding in each step all so far new documents to the initial corpus and update the initial topic model resulting in new topic probability distributions for all documents in the corpus, and

(iv) for FIGS, using the initial topic model for estimating a document-topic probability distribution distribution for new documents without changing the topic distribution of all other documents in the corpus.

LDA and both OLDA variants have similar held-out perplexity and the performance of both strategies is better than for FIGS.

9.3.3 Runtime

Now, we take a look at the runtime of the different strategies which is an important property. Figure 9.5 presents results for comparing the runtime of the different strategies considering the following three corpora differing in their initial corpus size: (i) small corpus containing 4k documents, (ii) medium corpus containing 8k documents, and (iii) large corpus containing 18k documents.

We analyze the runtime for each strategy adding three batches to each of the three different corpora. In case of LDA, we calculate four topic models for each size of the corpus, i.e., one initial topic model and three additional topic models, where each topic model is generated after adding one batch of corpus-extending documents to the corpus.

Using FIGS requires only one initial topic model and no additional models, since FIGS uses the initial topic model to infer the document-topic probability distribution $\theta_{d'_i}$ for each corpus-extending document in the batch of size T, represented by $\{d'_i\}_{i=0}^{T}$. For OLDA incr, we calculate one topic model from the initial corpus $\mathcal{D}$ and update the probability distributions of the model after adding the corpus-extending documents to $\mathcal{D}$ and go on working with the updated probability distributions of the topic model. For OLDA init, we always perform the update operation on a topic model representing the

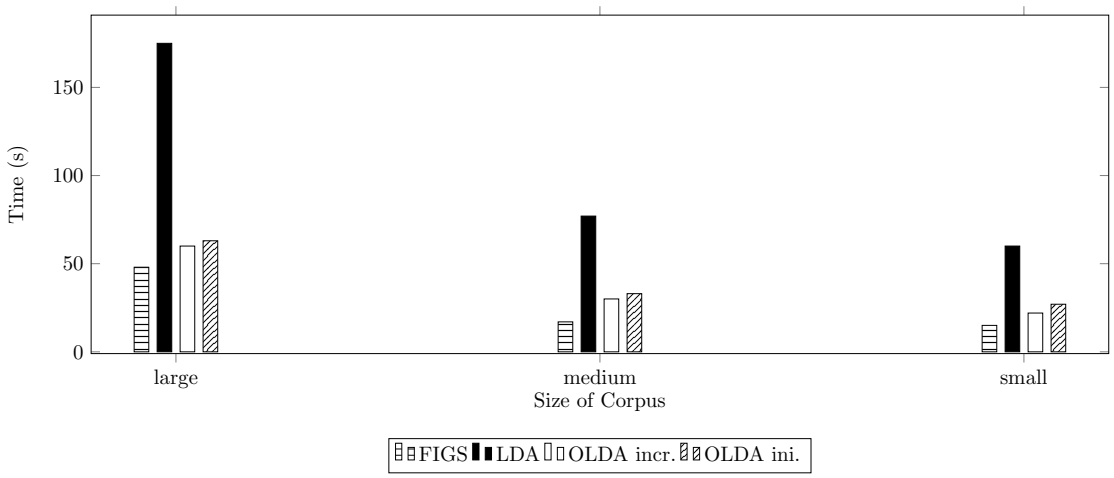


Figure 9.5: Runtime of FIGS, LDA, OLDA incr., and OLDA init. for large, medium, and small corpora.

initial corpus $\mathcal{D}$. Thus, in each step updating $\mathcal{D}$ with new documents we add the actual batch of documents and the previous batches to the initial corpus. Afterwards, we use OLDA to update the topic model to the documents in the extended corpus.

The runtime proportion between LDA, FIGS, OLDA incr. and OLDA init. is the same for all sizes of corpora. LDA has the largest runtime. FIGS is the fastest technique for each corpora. Both variants of OLDA have similar runtime performance.

9.3.4 Classification Performance

Next, we compare the classification performance between LDA, both variants of OLDA, and FIGS. Figure 9.6 present two plots of the classification performance. The initial corpus contains 8k documents. In both plots, eight batches of 1k documents are added to an initial corpus using one of the three strategies. Again, the left plot shows the classification performance for Type 1 and the right plot of Type 2. For Type 1 the initial corpus contains documents with content from different categories compared to the corpus-extending documents, and for Type 2 the initial corpus contains documents from the same categories as the corpus-extending documents.

In Figure 9.6 we can see that the FIGS strategy has the worst classification performance which results from non updating of the initial model within the FIGS strategy. For OLDA, both techniques (incr. and init.) start at nearly the same performance after adding the first batch of corpus-extending documents. Adding more batches to the initial corpus, the OLDA incr. becomes better while OLDA init. stays at the same classification performance. The adaptation process of OLDA indeed changes the document-topic probability distribution of documents which results in a better classification performance. The performance of OLDA incr. is better than for OLDA init.

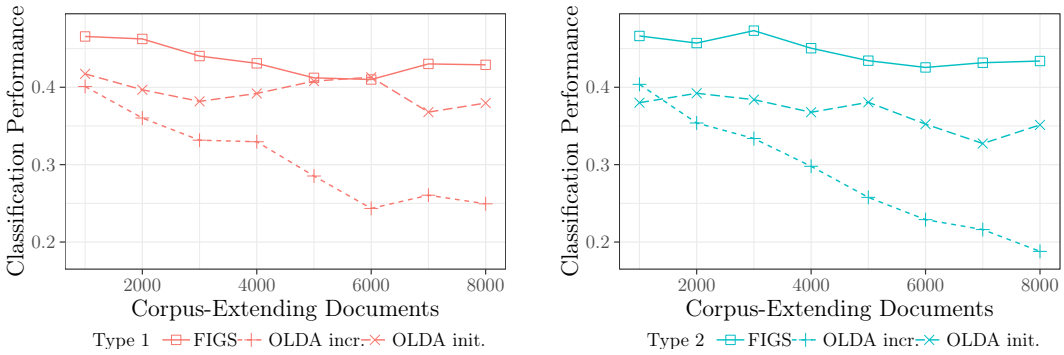


Figure 9.6: Classification performance of FIGS and both OLDA variants on corpora of Type 1 left and Type 2 right.

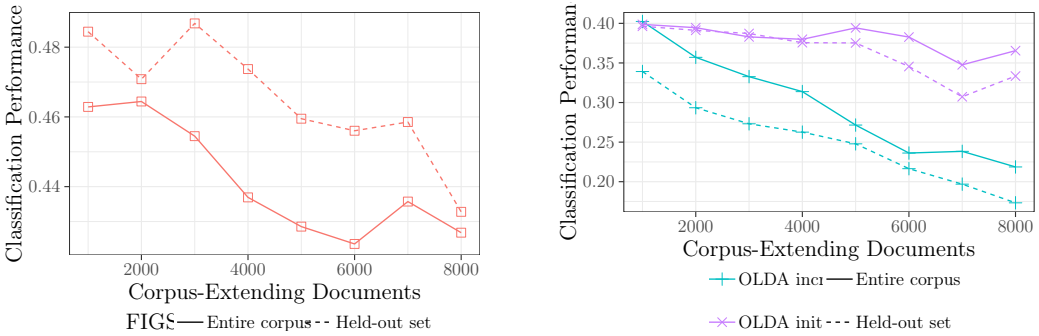


Figure 9.7: Classification performance of FIGS and OLDA on the entire corpus compared to the held-out set.

Generally, an agent is interested in comparing new documents with documents in a given corpus using document-topic probability distributions. Additionally, an agent might also be interested in the classification performance of documents which have been added to the corpus, besides the classification performance of FIGS and OLDA on the entire corpus. Figure 9.7 presents the classification performance on the initial documents and the batches of corpus-extending documents looking at the classification performance on the batches containing the documents extending the corpus. The left plot in Figure 9.7 shows the results for FIGS. As FIGS does not change the underlying model, the entire corpus reaches a better performance than the the held-out set. By adding more batches to the initial set of documents, both lines converge since the impact of the additional batches rise whereas they decrease for the initial documents. In the right plot of Figure 9.7, we present the results for OLDA on the entire corpus (solid line) and a held-out set (dashed lines). Additionally, we compare OLDA for the incremental variant (plus) and the initial variant (cross). In contrast to FIGS, the performance for OLDA incr. and OLDA init. is better on the held-out set than on the entire corpus and the classification performance

increases with an increasing number of documents extending to the initial corpus.

9.4 Interim Conclusion: Maintaining Topic Models

Topic modeling techniques can be used to estimate document-topic probability distributions and topic-word probability distribution for documents and topics in a corpus, respectively. We have evaluated three strategies to incorporate new documents for a given corpus. The agent might choose one of the three strategies depending on the task of an agent, since for some tasks it might be acceptable that the performance of the model decreases but a fast runtime is necessary. We have also compared different techniques to match the topic-word probability distribution generated from different topic models. In the context of our evaluation, we have evaluated the performance of LDA, FIGS, and two variants of OLDA regarding the perplexity of the documents in the corpus, the held-out set perplexity, the classification performance, and the runtime for corpora of different size. In conclusion, each of the three strategies has its advantages and disadvantages and the best strategy for estimating a topic-word probability distribution for corpus-extending documents depends on the individual task of an agent. Calculating a new topic model from all documents in an extended corpus is time-consuming but allows for an easy comparison between topic probability distributions of documents. OLDA allows for efficiently updating the initial topic model, accepting an increased corpus perplexity. FIGS is fast in estimating a topic probability distribution for new documents without changing the topic probability distribution of the documents in the corpus. The evaluation shows that for a small set of corpus-extending documents is might be sufficient to use the FIGS strategy estimating document-topic probability distribution for corpus-extending documents.

Chapter 10

Enhancing Relational Topic Models with Named-Entity Induced Links

In this chapter, we analyze the topic modeling performance of the Relational Topic modeling approach (RTM) considering NEs to create links between documents in a corpus. Generally, topic modeling approaches considering links between documents look at explicit link structures. However, we argue that the implicit link structure between documents might increase the performance of a topic model since the modeling approach considers hidden relations between documents. Instead of introducing a new topic model, we use the RTM approach, introduced by Chang and Blei (2009), and take advantage of the available linking function within the model to create links between documents. For all documents in a corpus, we add a link between two documents if the two documents share at least one NE. We analyze the topic modeling performance on three datasets, where each dataset contains a different number of links between the documents.

The following paper presented Named-Entity induced links to enhance relational topic models:

> Felix Kuhr, Matthis Lichtenberger, Tanya Braun, and Ralf Möller. Enhancing relational topic models with named entity induced links. In *IEEE 15th International Conference on Semantic Computing, ICSC 2021, Virtual, January 27-29, 2021*, pages 451–458, 2021

As previously described, LDA allows for modeling topics of a corpus of documents, assuming documents to be a mixture of topics, which determine the words of them. Various topic modeling approaches have been introduced that extend the well-known LDA approach. The RTM approach extends the LDA approach by modeling links for each pair of documents in a corpus as a binary random variable that is conditioned on the content of both documents. For each possible connection between all documents in $\mathcal{D}$ the RTM contains a link variable, and the number of link variables is given by $|\mathcal{D}|^2$. The model can be used to summarize documents in a corpus, predict links between documents, and predict words within documents, since it is a generative model. Again, in the RTM approach topics are represented as word probability distributions over the fixed vocabulary $\mathcal{V}$ generated from all words of documents in corpus $\mathcal{D}$. For further details, please refer to Chang and Blei (2009).

Given an explicit link structure, RTM can show a better model perplexity compared to a topic model like LDA, which does not consider any links. Generally, the link structure may come from a citation network of a corpus of articles, from a network of hyperlinks in a corpus of web pages, or from a social network of friends in a corpus of postings etc. and depends on the documents. However, in this chapter the link structure for RTM come from extractable NEs. Next, we present techniques to compare documents based on their document-topic probability distribution.

Considering an implicit link structure available from extractable NEs shared between documents, instead of creating links from explicit metadata like citations or hyperlinks. We denote this approach as NE-RTM as RTM works with implicit links via NEs. Over the last years, NEs have been used to improve the quality of discovered topics (Krasnashchok and Jouili (2018)) or entity prediction (Hu *et al.* (2013)). The underlying assumption is that documents in a corpus share topics if documents share NEs, e.g., references to the same person, place, or company.

To evaluate the performance of NE-RTM two new datasets are collected from the free online encyclopedia *Simple Wikipedia* and *The New York Times* newspaper. Both datasets have an explicit link structure for RTM through hyperlinks as well as a plethora of NEs to induce links for NE-RTM. We analyze the linking properties of NEs, their NE categories – e.g. person or city, and their perceived relevance to answer the question of which type of NEs should be considered for introducing links between documents within a corpus. Comparing the prediction accuracy of NE-RTM using different types of NE-induced links, the results show that additional links can increase the topic modeling performance. In summary, the contribution of this chapter is two-fold:

(i) Analysis of the properties of NEs improving the performance of the topic model and

(ii) comparison of the performance between LDA, RTM, and NE-RTM with different settings for NE-induced links.

The generative process of RTM consists of two parts. The first part is identical to the generative process of LDA. The second part concerns the links between documents. Intuitively, documents sharing a similar document-topic probability distribution should be linked together. A binary variable models the linking between documents, one variable for each pair of documents. The linking is defined as follows.

For each pair of documents $d, d' \in \mathcal{D}$:

$$\text{Draw binary link indicator: } y \mid z_d, z_{d'} \sim \psi(y = 1 \mid z_d, z_{d'}),$$

where function ψ estimates the link probability between d and d' and is defined by:

$$\psi(y = 1) = \iota(\boldsymbol{\eta}^T(\overline{\boldsymbol{z}}_d \circ \overline{\boldsymbol{z}}_{d'}) + \nu$$

where $\overline{z}_d = \frac{1}{N_d} \sum_n z_{d,n}$. The $\circ$ symbol denotes the Hadamad (element-wise) product. As in Chang and Blei (2009), function ι can be the sigmoid function or alternatively, the exponential mean function. If ι is the sigmoid function, it models each binary variable as a logistic regression with hidden covariates, parameterized by coefficients η and intercept ν. The covariates are constructed by the Hadamard product of $\overline{z}_d$ and $\overline{z}_{d'}$, modeling the similarity between the topics of d and d'. In case ι is the exponential mean function, the probabilities returned increase exponentially. RTM contains for each possible connection between all documents in $\mathcal{D}$ a link variable, and the number of link variables is given by $|\mathcal{D}|^2$. In summary, RTM defines a joint distribution over the words in each document and the links between them.

10.1 Evaluation of NE-RTM

We extend RTM by adding links between two documents if the documents have at least one NE in common. The assumption is that documents sharing NEs have similar topics since the same NEs appear in the text. We evaluate the performance using the perplexity measure on held-out data (see Newman *et al.* (2009) for details). We fit the parameters in NE-RTM by performing Gibbs sampling, i.e., the word likelihood is given by:

$$\log\left(p(w)\right) = \sum_{d=1}^{|\mathcal{D}|} \sum_{j=1}^{|\mathcal{V}|} n^{(jd)} \log\left(\sum_{k=1}^{K} \theta_{d,k}\beta_{j,k}\right)$$

10.1.1 Datasets

Datasets commonly used for evaluating topic models are *Cora* (McCallum *et al.* (2000)), *WebKB* (Craven *et al.* (1998)), or the Proceedings of the National Academy of Science (PNAS), where citations or hyperlinks determine links. However, these datasets contain only very few NEs and are therefore not suitable for evaluating NE-RTM. Thus, we generate three new datasets to evaluate the performance of RTM-NE:

- *wiki-sparse-500*: 500 documents randomly selected from the English Simple Wikipedia. An average article consists of 348 words (241 without stop words). Each document has only few links to other documents.
- *wiki-dense-500*: 500 documents from the English Simple Wikipedia containing many documents about countries. An average article consists of 692 words (466 without stop words). Documents have a high number of links to other documents.
- *nyt-500*: 500 articles from The New York Times. An average article consists of 922 words (619 without stop words). Each document has links to articles containing related coverage leading to 942 links in total.

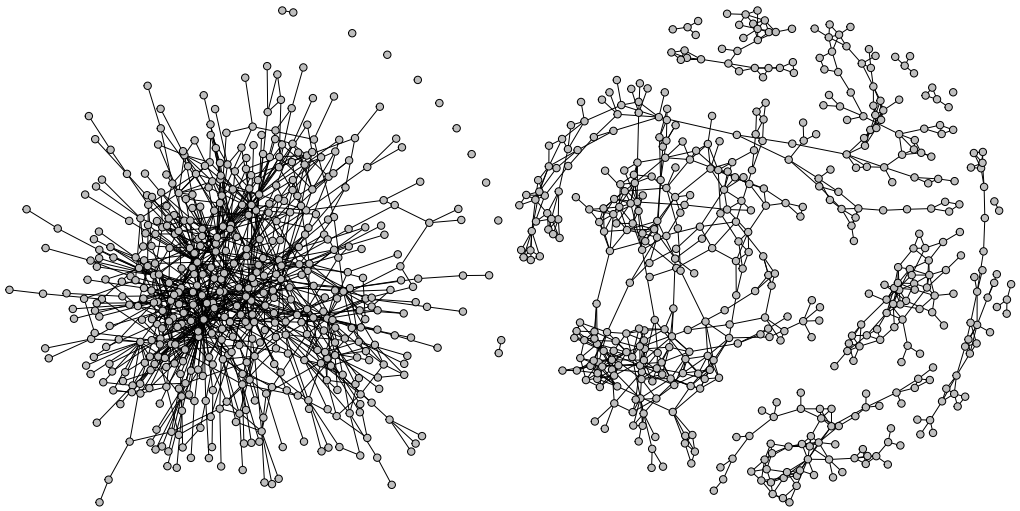


Figure 10.1: Explicit network structure of *wiki-sparse-500* (left) and *nyt-500* using hyperlinks and related coverage links from the New York Times between documents, respectively.

We preprocess all documents by (i) eliminating stop words, (ii) performing word stemming, and (iii) tokenizing the text. Simple English Wikipedia aims to restrict itself to use simple words and grammar, aimed at children and adults who are learning the language. The use of simple language comes in handy for our purposes, as there are not as many words of low frequency in the data. For documents in the *nyt-500* dataset, there are only few hyperlinks between documents. But there exists *related coverage* for most articles, containing links to up to five related articles. We have used the *related coverage* to generate the ground truth of links between documents.

Figure 10.1 present the network structure for documents in the *wiki-sparse-500* and *nyt-500* dataset. Figure 10.2 contains the corresponding node degree histograms for the network structure of all three datasets.

10.1.2 Categories of NEs

We use Open Calais (**?**) to extract NEs from datasets and store the available relevance value and category for each NE. Open Calais provides many categories besides typical categories resulting in 36 different categories in the three datasets. Tables 10.1 to 10.3 present for all three datasets the top 10 NE categories (by links) containing the total number of occurrences, the number of unique names of that type (bucket), and the number of links that could be created from the corresponding NEs.

In the *wiki* datasets, categories *country*, *industry-term*, *position*, and *continent* lead to most of the links between documents. The links from categories *organization* and *company* lead to only 10% of the number of links using categories *country*, *continent*,

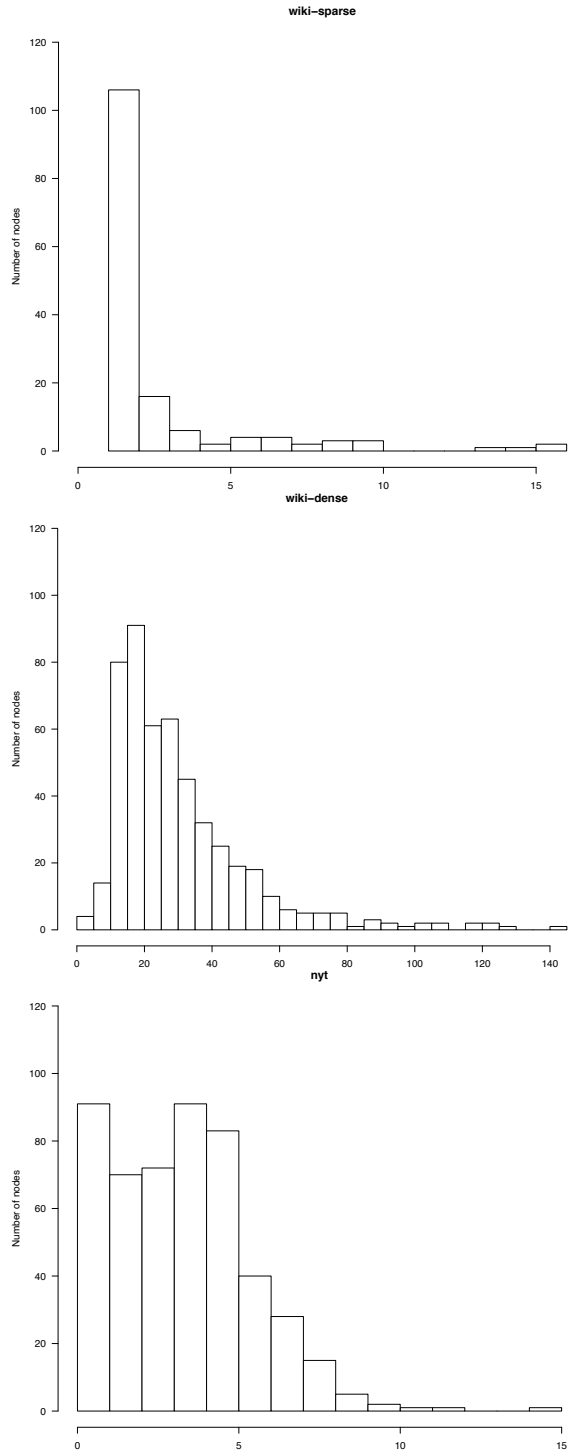


Figure 10.2: Node degree histograms of document network graphs for all datasets.

position, and *industry-term*. The *nyt-500* dataset also features categories *country* and *position* as the top two categories. However, categories *continent* and *industry-term* are only on the 8th and 9th position, respectively.

We consider different sets of NE-induced links, (i) all categories (all), (ii) only *organization* and *company* (org), and (iii) all categories excluding *country*, *continent*, *position*, and *industry-term* (exc), to test the effect of the number of links and different categories introduced for each dataset.

Table 10.1: Extractable NE categories in *wiki-sparse-500*.

#	category	total	bucket	links
1	Country	827	174	8717
2	IndustryTerm	664	486	1480
3	Position	843	479	1285
4	Continent	105	8	941
5	City	392	238	542
6	ProvinceOrState	205	91	353
7	Organization	456	368	217
8	Technology	262	178	198
9	MedicalCondition	275	219	112
10	Company	316	275	109

Table 10.2: Extractable NE categories in *wiki-dense-500*.

#	category	total	bucket	links
1	Country	3006	257	55011
2	Continent	353	8	11403
3	Position	1128	458	7676
4	IndustryTerm	1139	691	6202
5	Organization	1401	897	4457
6	Region	703	270	3569
7	ProvinceOrState	358	149	2760
8	SportsEvent	235	119	2636
9	City	1003	556	2452
10	NaturalFeature	851	476	1604

We also consider the following three thresholds for the relevance value r of NEs:

(i) all NEs, ignoring the relevance value (all),

(ii) only NEs having a relevance value ($r > 0$), and

Table 10.3: Extractable NE categories in *nyt-500*.

#	category	total	bucket	links
1	Country	1544	110	92854
2	Position	3406	1247	47732
3	City	1191	274	35537
4	Organization	2712	974	26512
5	Person	3763	2134	18508
6	Company	1681	815	9009
7	ProvinceOrState	724	101	6805
8	Continent	195	7	5155
9	IndustryTerm	1632	1057	3563
10	PublishedMedium	218	80	2400

(iii) only NEs having a relevance value above 0.2 ($r > 0.2$).

Next, we present results for NE-induced links for the three different sets of categories and relevance thresholds.

10.1.3 Empirical Results

As RTM is a predictive model, we can measure the performance by withholding data in the training phase and examining the likelihood of predicting the held-out ground truth in a typical machine learning fashion. We generate the held-out set by removing half of the words from 10% of documents. After we have fitted the model, the topic probabilities for each document and the word distribution for each topic are known, which allows us to determine the predicted probability of the held-out data.

In Table 10.4, we present the perplexities for all three datasets using the different settings for NE induced links for the Relational Topic Model as described above and the following parameter setting: $k = 15$, $\alpha = 0.1$, $\beta = 0.1$, and $\eta = 0.1$ Additionally, we provide a baseline for the perplexity using the LDA topic model without any links (first row) and a basic RTM using only explicit links (second row). Explicit links are hyperlinks for Wikipedia documents and related coverage references for The New York Times articles. Rows 3 to 11 contain nine different NE-RTM settings. The bold values show the best perplexity for each dataset. The italic values represent a perplexity lower than the perplexity possible with basic RTM. The perplexity of RTM is better than the perplexity of LDA for all datasets. Additionally, as we have expected, the performance of NE-RTM is better than the perplexity of RTM using only explicit links. NE-induced links can enhance the performance of topic models, but simply adding links between documents sharing NEs does not automatically improve the performance and the choice of NE categories needs to be considered. Thus, it is no surprise that for all three datasets, the best perplexity is given by different configurations, since each dataset represents a

Table 10.4: Topic model performance.

Topic Model	Links	Relevance	dense	sparse	nyt
LDA			1208.32	1156.91	1576.78
RTM			1196.16	1126.89	1573.71
NE-RTM	org	all	1193.89	1124.97	**1554.80**
NE-RTM	org	$r > 0$	1195.56	1125.47	1554.91
NE-RTM	org	$r > 0.2$	1195.26	*1127.19*	1570.68
NE-RTM	exc	all	1192.64	1121.04	*1625.36*
NE-RTM	exc	$r > 0$	1191.45	**1120.20**	*1621.90*
NE-RTM	exc	$r > 0.2$	1194.03	*1128.41*	1567.62
NE-RTM	all	all	1172.22	1207.44	*1824.66*
NE-RTM	all	$r > 0$	**1169.37**	1207.94	*1822.15*
NE-RTM	all	$r > 0.2$	1194.00	*1128.61*	*1576.14*

different characteristic.

Overall, the dataset with most links, *wiki-dense-500*, benefits the most from added links through NEs. The best perplexity is given by considering all entities. The *nyt-500* dataset mostly benefits from the (org) setup while (all) leads to worse results than basic RTM. One reason for this behaviour might be, that only articles about the same company contain similar text, while documents containing NEs like countries, continents, or positions appear often in different coherencies within a newspaper. The *wiki-sparse-500* dataset benefits in 2 out of 3 cases but exhibits the smallest change in perplexity compared to the other two sets. The best perplexity is given by ignoring *countries*, *continents*, *positions*, and *industry-terms*. The *wiki-dense-500* dataset shows the largest improvement. The dataset which is linked to a high degree from the start seems to win the most by adding further links. The links induced by NEs from organizations increase the perplexity for all datasets except with the *wiki-sparse-500* dataset and $r > 0.2$.

10.2 Interim Conclusion: Named-Entity Induced Links

RTM uses explicit links between documents to improve the performance of a topic model for a corpus of documents, where the performance is measured by using the model perplexity. In this chapter, we have generated links between documents based on matching NE and have analysed the performance model perplexity compared to LDA and RTM. The results on three datasets show that the performance of RTM can be increased by adding links originating from NER. The performance depends on the documents in the corpus and the NEs used to induce links. Future work might include exploring more diverse data sets and analyzing the effect of different NE recognizers.

Chapter 11

Conclusion

Manually annotating documents is a major task in corpus linguistics and estimating valuable data that can be associated with documents in a corpus is an important discipline. In contrast to available annotation systems, this dissertation lay the foundation for automatically enriching documents in a corpus with context-aware and subjective content descriptions for documents by considering the individual collection of documents to support an information retrieval agent on its corpus-specific tasks, e.g., identifying a set of documents in a corpus being similar to a new document or estimating SCDs for a new document based on the SCDs associated with documents available in a corpus.

We summarize the contributions that this dissertation has made towards subjective content descriptions and provide some directions for future work.

11.1 Summary of Contributions

In this dissertation, we focus on associating subjective and context-aware content descriptions to documents in a corpus. We combine different similarities to automatically enrich documents in a corpus with subjective content descriptions and present different approaches to enrich documents with subjective content descriptions for various scenarios. In this dissertation, we make a number of contributions to subjective content descriptions and can summarize the contributions as follows:

Context-specific corpus enrichment Since manually generating SCDs and associating SCDs with documents is a time-consuming and expensive tasks. We introduce (i) a definition of a holistic (D-similarity) and symbolic similarity (T-similarity) between documents of the same corpus, resulting in (ii) a definition of a corpus-driven relevance value for SCDs representing the relevance of an SCD for the content of a document, and (iii) an unsupervised algorithm using a combination of both similarity values and the relevance value to enrich sparsely and weakly annotated documents with SCDs of related documents from the same corpus supporting an agent in its tasks.

Additionally, we show the potential of augmenting and automatically enriching a corpus with new documents while considering the context is explicitly represented by the

corpus-driven topic-word probability distribution. Estimating the category of a new document based on the individual collection of documents in a corpus is non-trivial. Thus, we use a combination of different HMMs to estimate a predefined category for new documents by analyzing the most likely sequence of MPSCD similarity values.

Identifying SCDs among Texts We have defined the inline SCD problem, where words of SCDs are interleaved with the content of documents. Additionally, we present an approach to identify textual SCDs among the usual text of documents by using the SCD-word probability distribution which encodes the most likely words to occur with an SCD, as well as an HMM, which encodes being part of an SCD or not as the hidden state. Calculating a most probable sequence of hidden states in the HMM then allows for identifying the most probable windows for inline SCDs.

Context-aware adaptation of SCD-word probability distributions SCDs associated with documents in one corpus might add a value for the tasks of an agent working with documents in another corpus. Automatically associating SCDs to documents in one corpus that are associated with documents in another corpus is non-trivial while considering the context-shift between both corpora. Thus, we introduce an unsupervised domain adaptation algorithm adapting the SCD-word probability distribution from a source corpus to a destination corpus supporting an agent in its tasks with an initial set of context-aware SCDs for documents in a non-annotated target corpus. The reusability of SCDs between different corpora is an important property since manually generating corpus-specific new SCDs is a time-consuming and expensive task.

Maintaining topic models for growing corpora The individual collection of documents in a corpus might change over time, since an agent might decide to extend its corpus with additional documents. Changing the individual collection of documents results in another corpus and requires an adaptation of the similarity between documents as well as a modified relevance value of each SCD. Thus, we analyze different techniques to incorporate corpus-extending documents into a given topic model resulting from the initial documents in a corpus s.t. an agent can identify a set of similar documents in the corpus for the corpus-extending documents based on the documents' topic similarity.

Named Entity induced links for relational topic models Relations between documents provide important data to optimize the performance of topic models. The text of similar documents might share entities and we assume that documents sharing NEs are somehow connected to each other. We use the well-known RTM approach to represent the NE-based relationship between documents in a corpus by adding a link between two documents sharing a NE. Additionally, we estimate the properties of NEs improving

the performance of the RTM and compare the performance between LDA and RTM by considering different settings for NE induced links.

11.2 Future Work

Moving forward from this dissertation, there are a variety of ways to continue research in context-aware subjective content descriptions for documents. We select the following two topics that might extend this dissertation.

Dropping out Documents Generally, the size of a corpus is not fixed and an agent might extend a corpus with new documents or remove some documents from the corpus for various reasons. In this dissertation, we have analyzed the performance of different techniques for documents extending a corpus. However, if an agent detects a document containing *fake-news* or unimportant content for its task, the agent might remove this document from its corpus. In Chapter 5, we have introduced an algorithm iteratively enriching sparsely and weakly annotated documents with SCDs that are associated with other documents in the same corpus. Generally, the iterative algorithm associates SCDs with documents based on a relevance value of an SCD. The relevance value depends on the similarity between SCDs associated with documents and documents. If documents drop out of the corpus, we might have to update the automatically enriched SCDs, e.g., if documents contain fake-news, the associated SCDs might contain wrong data. So, introducing some kind of rollback mechanisms for the iterative algorithm handling dropping out documents would be a valuable extension of this dissertation.

SCD-based Language Model In this dissertation, we assume that a corpus is available and the content of documents is accessible. We have performed different operations on SCDs and documents in a given corpus. However, in some scenarios, it is not possible to make documents in a corpus publicly available — one well-known reason is the copyright of a document. Thus, an extension of the SCD-word probability distribution approach is generating a new language model to estimate the MPSCDs for a new document without sharing the individual collection of documents in a corpus, but only the model. Generally, the language model should embed documents of a corpus into an embedding space s.t. the influence of SCD-word probability distributions can be represented based on proximity making the original documents in a corpus not needed. The language model might be a parameterized language model to include parameters like the origin, the county, or the language of documents.

Part III

Appendix

Appendix A

Additional Results for T- and D-similarity

A.1 SCD-Enrichment between Documents focusing on T-Similarity

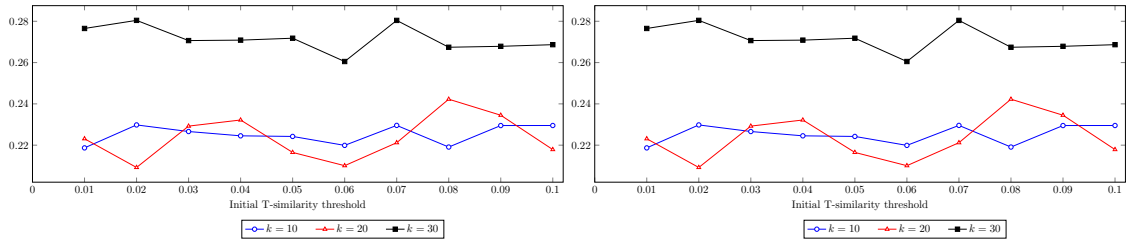


Figure A.1: Evaluating different T-similarity values for dataset 1 (top) and dataset 2 (bottom).

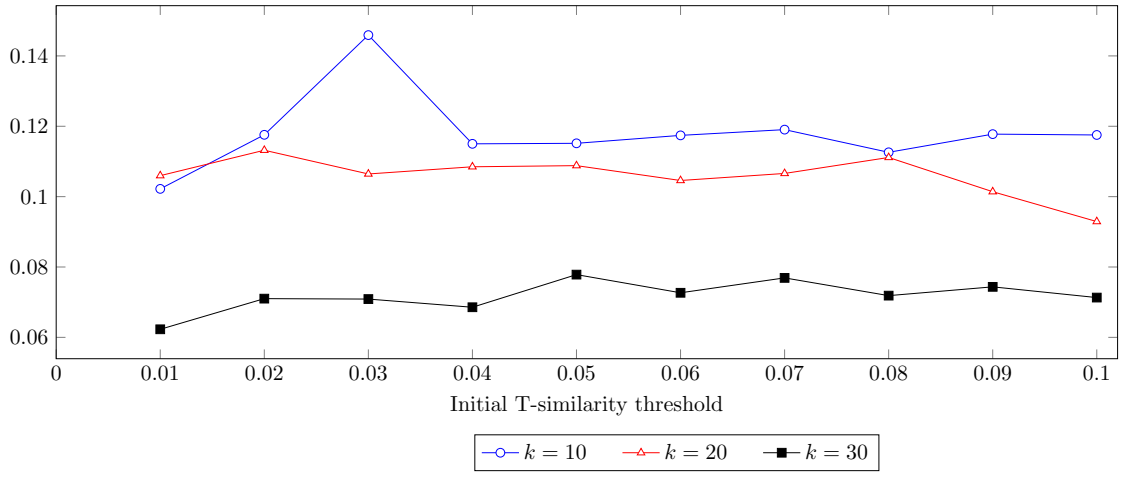


Figure A.2: Evaluating different T-similarity values for dataset 3.

A.2 SCD-Enrichment between Documents focusing on D-Similarity

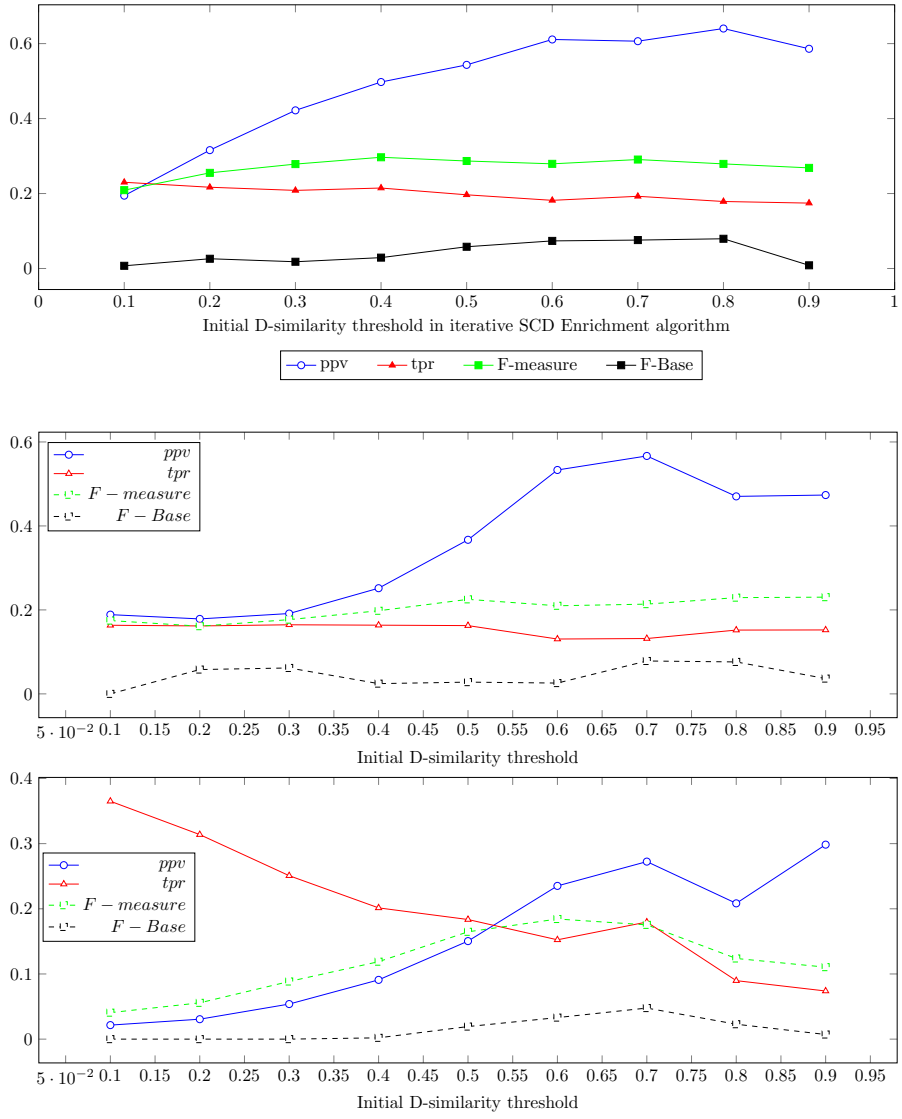


Figure A.3: Performance depending on D-similarity; $k = 30$, $r = 0.90$, $\epsilon = 0.01$, $\overline{Sim_{G^{d_e}}} = 0.1$. F-Base represents the F_1-score for randomly guessing possible SCDs for SCD sets.

A.3 Number of Iterations in Algorithm 5

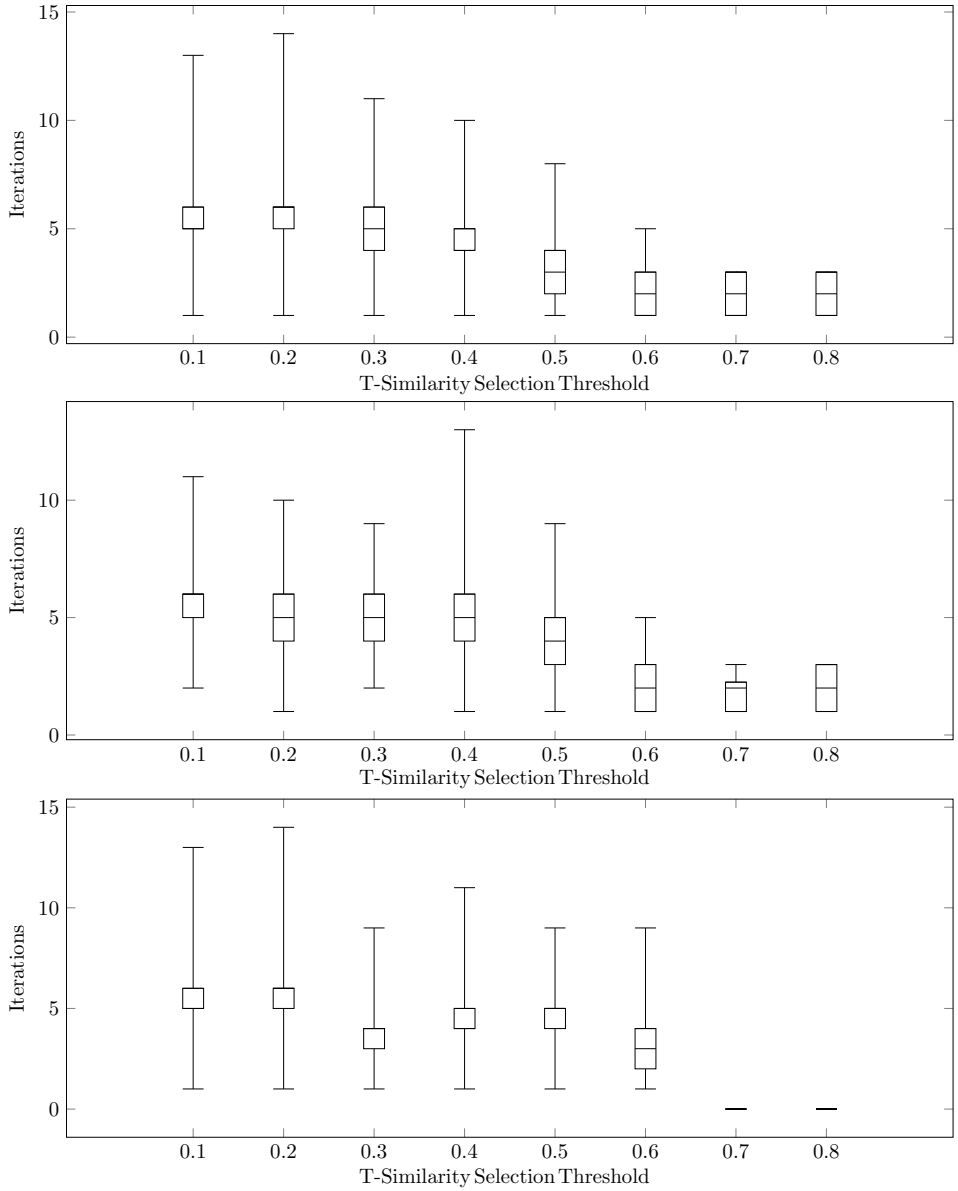


Figure A.4: Number of necessary iterations for dataset 1 (left), dataset 2 (center), and dataset 3 (right) until Algorithm 5 reaches a fixed point, depending on Sim_T.

Appendix B

Datasets

We use three datasets in the empirical evaluation in Section 5.3. To receive the text it is necessary to insert the name of a document in the following url: `https://en.wikipedia.org/wiki/<name>`. Example: The text of document `BMW Z4` is given by the url: `https://en.wikipedia.org/wiki/BMW_Z4`.

The following three sections represent the documents for each dataset used in Section 5.3.

B.1 Dataset 1: BMW

BMW Z4, BMW X5 (F15), BMW Z4 (E89), BMW M2B15, BMW 7 Series (F01), BMW 8 Series, BMW M30, BMW Z3, BMW M54, BMW N42, BMW N62, BMW 1 Series, BMW X4, BMW 3200 CS, BMW 321, BMW M78, BMW 3 Series, BMW 501, BMW Z8, BMW E9, BMW M21, BMW M41, BMW M57, BMW X3, BMW Z1, BMW N47, BMW B47, BMW M328, BMW i8, BMW M50, BMW S14, BMW 303, BMW X1, BMW B58, BMW X-Coupe, BMW Dixi, BMW S85, BMW 600, BMW 700, BMW Turbo, BMW GINA, BMW 327, BMW 503, BMW R100, BMW M6, BMW M Roadster, BMW P60B40, BMW X7, BMW 801, BMW X, BMW i

B.2 Dataset 2: Mercedes

Unimog 435, Mercedes-Benz W126, Mercedes-Maybach 6, Mercedes-Benz Ponton, Mercedes-Benz W111, Mercedes-Benz W112, Mercedes-Benz W136, Mercedes-Benz W186, Mercedes-Benz MB100, Mercedes-Benz TN, Mercedes-Benz Citan, Mercedes-Benz W128, Mercedes-Benz W105, Mercedes-Benz Bionic, Mercedes-Benz Atego, Mercedes-Benz M110 engine, Mercedes-Benz OM617, Mercedes-Benz SL-Class (R129), Mercedes-Benz W108, Mercedes-Benz W116, Mercedes-Benz W124, Mercedes-Benz W125 1937 4, Mercedes-Benz OM601 engine, Mercedes-Benz W154 1938 3, Mercedes-Benz 320A, Mercedes-Benz 380 (1933), Mercedes-Benz 600, Mercedes-Benz CLK-Class, Mercedes-Benz Vito, Mercedes-Benz OM605 engine, Mercedes-Benz W25, Mercedes-Benz C11, Mercedes-Benz W165, Mercedes-Benz W125, Mercedes-Benz W154, Benz Bz.IV, Mercedes D.I,

Mercedes-Benz AMG C-Class DTM (W204), Mercedes-Benz W25, Mercedes-Benz W125, Mercedes-Benz M110 engine, Mercedes-Benz E-Class (W210), Mercedes-Benz SLK-Class (R170), Mercedes-Benz W188, Mercedes-Benz W189, Mercedes-Benz MB100, Mercedes-Benz TN, Mercedes-Benz OM654 engine, Mercedes-Benz SK, Mercedes-Benz NG

B.3 Dataset 3: US universities

Adams State University, Alabama Agricultural and Mechanical University, Alabama State University, Albany State University, Alcorn State University, Alfred State College, Alfred University, American University, Angelo State University, Appalachian State University, Arizona State University, Arizona State University Polytechnic campus, Arizona State University Tempe campus, Arizona State University West campus, Arkansas State University, Arkansas Tech University, Armstrong State University, Athens State University, Atlanta Metropolitan State College, Auburn University, Auburn University at Montgomery, Augusta University, Austin Peay State University, Bainbridge State College, Ball State University, Bemidji State University, Binghamton University, Black Hills State University, Bloomsburg University of Pennsylvania, Bluefield State College, Boise State University, Bowie State University, Bowling Green State University, Bridgewater State University, Brooklyn College, Buffalo State College, Ohio State University Agricultural Technical Institute, Peru State College, Massachusetts College of Art and Design, Iowa State University, Kutztown University of Pennsylvania, Purdue University, Northwest Missouri State University, Emporia State University, Morgan State University, Northern Kentucky University, Rowan University, Cornell University, Eastern New Mexico University, Southern University at New Orleans

Appendix C

Datasets 2

We use three datasets in the empirical evaluation in Section 7.4. The following three sections represent the documents for each dataset. To receive the text it is necessary to insert the name of a document in the following url: `https://en.wikipedia.org/wiki/<name>`. The text of document George Washington is given by the url: `https://en.wikipedia.org/wiki/George_Washington`.

C.1 Dataset 1: U.S. Presidents

George Washington, John Adams, Thomas Jefferson, James Madison, James Monroe, John Quincy Adams, Andrew Jackson, Martin Van Buren, William Henry Harrison, John Tyler, James K. Polk, Zachary Taylor, Millard Fillmore, Franklin Pierce, James Buchanan, Abraham Lincoln, Andrew Johnson, Ulysses S. Grant, Rutherford B. Hayes, James A. Garfield, Chester A. Arthur, Grover Cleveland, Benjamin Harrison, Grover Cleveland, William McKinley, William Howard Taft, Woodrow Wilson, Warren G. Harding, Calvin Coolidge, Herbert Hoover, Franklin D. Roosevelt, Harry S. Truman, Dwight D. Eisenhower, John F. Kennedy, Lyndon B. Johnson, Richard Nixon, Gerald Ford, Jimmy Carter, Ronald Reagan, George H. W. Bush, Bill Clinton, George W. Bush, Barack Obama

C.2 Dataset 2: U.K. Prime Ministers

Sir Robert Walpole, Spencer Compton, Henry Pelham, Thomas Pelham-Holles, William Cavendish, Thomas Pelham-Holles, John Stuart, George Grenville, Charles Watson-Wentworth, William Pitt the Elder, Augustus FitzRoy, Frederick North, William Petty, William Cavendish-Bentinck, William Pitt the Younger, Henry Addington, William Grenville, William Cavendish-Bentinck, Spencer Perceval, Robert Jenkinson, George Canning, Frederick John Robinson, Arthur Wellesley, Charles Grey, William Lamb, Sir Robert Peel, Lord John Russell, Edward Smith-Stanley, George Hamilton-Gordon, Henry John Temple, John Russell, Benjamin Disraeli, William Ewart Gladstone, Robert Gascoyne-Cecil, Archibald Primrose, Arthur Balfour, Sir Henry Campbell-Bannerman,

H. H. Asquith, David Lloyd George, Bonar Law, Stanley Baldwin, Ramsay MacDonald, Stanley Baldwin, Neville Chamberlain, Winston Churchill, Clement Attlee, Sir Anthony Eden, Harold Macmillan, Sir Alec Douglas-Home, Harold Wilson, Edward Heath, James Callaghan, Margaret Thatcher, John Major, Tony Blair, Gordon Brown, David Cameron, Theresa May

Bibliography

Rie Kubota Ando and Tong Zhang. A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research*, 6(Nov):1817–1853, 2005.

Gabor Angeli, Melvin Johnson Premkumar, and Christopher D Manning. Leveraging linguistic structure for open domain information extraction. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics (ACL 2015)*, 2015.

Leonard E Baum, Ted Petrie, George Soules, and Norman Weiss. A maximization technique occurring in the statistical analysis of probabilistic functions of markov chains. *The annals of mathematical statistics*, 41(1):164–171, 1970.

Sean Bechhofer, Leslie Carr, Carole Goble, Simon Kampa, and Timothy Miles-Board. The semantics of semantic annotation. In *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*, pages 1152–1167. Springer, 2002.

Immanuel Bekker et al. Aristotelis opera edidit academia regia borussica. *Volumen primum-Volumen alterum*, 1831.

Magnus Bender, Tanya Braun, Marcel Gehrke, Felix Kuhr, Ralf Möller, and Simon Schiff. Identifying subjective content descriptions among text. In *IEEE 15th International Conference on Semantic Computing, ICSC 2021, Virtual, January 27-29, 2021*, pages 451–458, 2021.

Magnus Bender, Tanya Braun, Marcel Gehrke, Felix Kuhr, Ralf Möller, and Simon Schiff. Identifying subjective content descriptions among text. *will be published in: Int. J. Semantic Comput.*, 15(4), 2021.

Chris Biemann. Creating a system for lexical substitutions from scratch using crowdsourcing. *Language Resources and Evaluation*, 47(1):97–122, 2013.

David M. Blei and John D. Lafferty. Dynamic topic models. In *Machine Learning, Proceedings of the Twenty-Third International Conference (ICML 2006), Pittsburgh, Pennsylvania, USA, June 25-29, 2006*, pages 113–120, 2006.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

John Blitzer, Ryan McDonald, and Fernando Pereira. Domain adaptation with structural correspondence learning. In *Proceedings of the 2006 conference on empirical methods in natural language processing*, pages 120–128. Association for Computational Linguistics, 2006.

Kalina Bontcheva, Hamish Cunningham, Ian Roberts, Angus Roberts, Valentin Tablan, Niraj Aswani, and Genevieve Gorrell. GATE teamware: a web-based, collaborative text annotation framework. *Language Resources and Evaluation*, 47(4):1007–1029, 2013.

Kalina Bontcheva, Ian Roberts, Leon Derczynski, and Dominic Rout. The gate crowdsourcing plugin: Crowdsourcing annotated corpora made easy. In *Proceedings of the Demonstrations at the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 97–100, 2014.

Tanya Braun, Felix Kuhr, and Ralf Möller. Unsupervised text annotations. In *Formal and Cognitive Reasoning - Workshop at the 40th Annual German Conference on AI (KI-2017)*, 2017.

Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R Hruschka Jr, and Tom M Mitchell. Toward an architecture for never-ending language learning. In *AAAI*, volume 5, 2010.

Jonathan Chang and David Blei. Relational topic models for document networks. In *Artificial Intelligence and Statistics*, pages 81–88, 2009.

Ciprian Chelba and Alex Acero. Adaptation of maximum entropy capitalizer: Little data can help a lot. *Computer Speech & Language*, 20(4):382–399, 2006.

Nancy A Chinchor. Overview of proceedings of the seventh message understanding conference (muc-7)/met-2. In *Proceedings of the Seventh Message Understanding Conference (MUC-7). Fairfax, VA. http://www. itl. nist. gov/iaui/894.02/related_ projects/muc*, 1998.

Mark Craven, Dan DiPasquo, Dayne Freitag, Andrew McCallum, Tom M. Mitchell, Kamal Nigam, and Seán Slattery. Learning to extract symbolic knowledge from the world wide web. In *Proc. of the Fifteenth National Conference on Artificial Intelligence and Tenth Innovative Applications of Artificial Intelligence Conference, AAAI 98, IAAI 98, July 26-30, 1998, Madison, Wisconsin, USA.*, pages 509–516, 1998.

Hamish Cunningham. Gate, a general architecture for text engineering. *Computers and the Humanities*, 36(2):223–254, 2002.

William M Darling. A theoretical and practical implementation tutorial on topic modeling and gibbs sampling. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies*, pages 642–647, 2011.

David Day, Alan Goldschen, and John Henderson. A framework for cross-document annotation. In *LREC*, 2000.

Scott Deerwester, Susan T Dumais, George W Furnas, Thomas K Landauer, and Richard Harshman. Indexing by latent semantic analysis. *Journal of the American society for information science*, 41(6):391–407, 1990.

Gerald DeJong. Prediction and substantiation: A new approach to natural language processing. *Cognitive Science*, 3(3):251–273, 1979.

Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society. Series B (methodological)*, pages 1–38, 1977.

Stephen Dill, Nadav Eiron, David Gibson, Daniel Gruhl, Ramanathan Guha, Anant Jhingran, Tapas Kanungo, Sridhar Rajagopalan, Andrew Tomkins, John A Tomlin, et al. Semtag and seeker: Bootstrapping the semantic web via automated semantic annotation. In *Proceedings of the 12th international conference on World Wide Web*, pages 178–186. ACM, 2003.

Xin Luna Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Kevin Murphy, Shaohua Sun, and Wei Zhang. From data fusion to knowledge fusion. *Proceedings of the VLDB Endowment*, 7(10):881–892, 2014.

Richard Eckart de Castilho, Éva Mújdricza-Maydt, Seid Muhie Yimam, Silvana Hartmann, Iryna Gurevych, Anette Frank, and Chris Biemann. A web-based tool for the integrated annotation of semantic and syntactic structures. In *Proceedings of the Workshop on Language Technology Resources and Tools for Digital Humanities (LT4DH)*, pages 76–84, Osaka, Japan, December 2016. The COLING 2016 Organizing Committee.

W. N. Francis and H. Kucera. Brown corpus manual. Technical report, Department of Linguistics, Brown University, Providence, Rhode Island, US, 1979.

Aldo Gangemi. A comparison of knowledge extraction tools for the semantic web. In *Extended Semantic Web Conference*, pages 351–366. Springer, 2013.

Stuart Geman and Donald Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Transactions on pattern analysis and machine intelligence*, 1(6):721–741, 1984.

Lise Getoor and Christopher P. Diehl. Link mining: a survey. *SIGKDD Explorations*, 7(2):3–12, 2005.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Domain adaptation for large-scale sentiment classification: A deep learning approach. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 513–520, 2011.

Thomas L Griffiths and Mark Steyvers. Finding scientific topics. *National academy of Sciences*, 101(suppl 1):5228–5235, 2004.

Tom Griffiths. Gibbs sampling in the generative model of latent dirichlet allocation. 2002.

Ralph Grishman and Beth M Sundheim. Message understanding conference-6: A brief history. In *COLING 1996 Volume 1: The 16th International Conference on Computational Linguistics*, 1996.

Ramanathan Guha and Rob McCool. Tap: A semantic web platform. *Computer Networks*, 42:2003, 2003.

Ernst Hellinger. Neue begründung der theorie quadratischer formen von unendlichvielen veränderlichen. *Journal für die reine und angewandte Mathematik*, 136:210–271, 1909. (in German).

Matthew D. Hoffman, David M. Blei, and Francis R. Bach. Online learning for latent dirichlet allocation. In *Advances in Neural Information Processing Systems 23: 24th Annual Conference on Neural Information Processing Systems 2010. Proceedings of a meeting held 6-9 December 2010, Vancouver, British Columbia, Canada.*, pages 856–864, 2010.

Thomas Hofmann. Probabilistic latent semantic indexing. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 50–57, 1999.

Linmei Hu, Juan-Zi Li, Zhihui Li, Chao Shao, and Zhixing Li. Incorporating entities in news topic modeling. In *Natural Language Processing and Chinese Computing - Second CCF Conference, NLPCC 2013, Chongqing, China, November 15-19, 2013, Proc.*, pages 139–150, 2013.

ISO-Standard:24612:2012. 24612: 2012 language resource management - linguistic annotation framework. *Project leader: Nancy Ide*, 2012.

David Jurgens. Embracing ambiguity: A comparison of annotation methodologies for crowdsourcing word sense labels. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 556–562, 2013.

Kanako Komiya, Shota Suzuki, Minoru Sasaki, Hiroyuki Shinnou, and Manabu Okumura. Domain adaptation for word sense disambiguation using word embeddings. In *International Conference on Computational Linguistics and Intelligent Text Processing*, pages 195–206. Springer, 2017.

Katsiaryna Krasnashchok and Salim Jouili. Improving topic quality by promoting named entities in topic modeling. In *Proc. of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 2: Short Papers*, pages 247–253, 2018.

Felix Kuhr and Ralf Möller. Constructing and maintaining corpus-driven annotations. In *2019 IEEE 13th International Conference on Semantic Computing (ICSC)*, pages 462–467, Jan 2019.

Felix Kuhr and Özgür L Özcep. Theory interpretations for topic models. In *The Thirty-Third International Flairs Conference*, 2020.

Felix Kuhr, Tanya Braun, Magnus Bender, and Ralf Möller. To Extend or not to Extend? Context-specific Corpus Enrichment. In *Proceedings of AI 2019: Advances in Artificial Intelligence*, pages 357–368. Springer, 2019.

Felix Kuhr, Bjarne Witten, and Ralf Möller. Corpus-driven annotation enrichment. In *2019 IEEE 13th International Conference on Semantic Computing (ICSC)*, pages 138–141. IEEE, 2019.

Felix Kuhr, Magnus Bender, Tanya Braun, and Ralf Möller. Context-specific adaptation of subjective content descriptions. In *Proceedings of AI 2020: Advances in Artificial Intelligence*. Springer, 2020.

Felix Kuhr, Magnus Bender, Tanya Braun, and Ralf Möller. Maintaining topic models for growing corpora. In *IEEE 14th International Conference on Semantic Computing, ICSC 2020, San Diego, CA, USA, February 3-5, 2020*, pages 451–458, 2020.

Felix Kuhr, Tanya Braun, Magnus Bender, and Ralf Möller. Augmenting and automating corpus enrichment. *Int. J. Semantic Comput.*, 14(2):173–197, 2020.

Felix Kuhr, Tanya Braun, and Ralf Möller. Augmenting and automating corpus enrichment. In *2020 IEEE 14th International Conference on Semantic Computing (ICSC)*, pages 69–76. IEEE, 2020.

Felix Kuhr, Magnus Bender, Tanya Braun, and Ralf Möller. Context-specific adaptation of subjective content descriptions. In *IEEE 15th International Conference on Semantic Computing, ICSC 2021, Virtual, January 27-29, 2021*, pages 451–458, 2021.

Felix Kuhr, Matthis Lichtenberger, Tanya Braun, and Ralf Möller. Enhancing relational topic models with named entity induced links. In *IEEE 15th International Conference on Semantic Computing, ICSC 2021, Virtual, January 27-29, 2021*, pages 451–458, 2021.

Solomon Kullback and Richard A Leibler. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86, 1951.

Ken Lang. Newsweeder: Learning to filter netnews. In *Machine Learning Proceedings 1995*, pages 331–339. Elsevier, 1995.

Geoffrey Leech. Adding linguistic annotation. 2005.

Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick van Kleef, Sören Auer, and Bizer Chris. DBpedia - a large-scale, multilingual knowledge base extracted from wikipedia. In *DBpedia - A Large-scale, Multilingual Knowledge Base Extracted from Wikipedia*. IOS Press, 2014.

Xiaofeng Liao and Zhiming Zhao. Unsupervised approaches for textual semantic annotation, a survey. *ACM Computing Surveys (CSUR)*, 52(4):1–45, 2019.

Yongxin Liao, Mario Lezoche, Hervé Panetto, and Nacer Boudjlida. Why, where and how to use semantic annotation for systems interoperability. In *1st UNITE Doctoral Symposium*, pages 71–78, 2011.

Dekang Lin and Xiaoyun Wu. Phrase clustering for discriminative learning. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 1030–1038. Association for Computational Linguistics, 2009.

Ian Marshall. Choice of grammatical word-class without global syntactic analysis: tagging words in the lob corpus. *Computers and the Humanities*, pages 139–150, 1983.

Luisa März, Dietrich Trautmann, and Benjamin Roth. Domain adaptation for part-of-speech tagging of noisy user-generated text. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3415–3420, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

Andrew McCallum, Kamal Nigam, Jason Rennie, and Kristie Seymore. Automating the construction of internet portals with machine learning. *Inf. Retr.*, 3(2):127–163, 2000.

Andrew Kachites McCallum. Mallet: A machine learning for language toolkit. http://mallet.cs.umass.edu, 2002.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. *arXiv preprint arXiv:1310.4546*, 2013.

Thomas Minka and John Lafferty. Expectation-propagation for the generative aspect model. In *Proceedings of the Eighteenth conference on Uncertainty in artificial intelligence*, pages 352–359. Morgan Kaufmann Publishers Inc., 2002.

David Newman, Arthur U. Asuncion, Padhraic Smyth, and Max Welling. Distributed algorithms for topic models. *Journal of Machine Learning Research*, 10:1801–1828, 2009.

Pedro Oliveira and João Rocha. Semantic annotation tools survey. In *Computational Intelligence and Data Mining (CIDM), 2013 IEEE Symposium on*, pages 301–307. IEEE, 2013.

Georgios Petasis, Vangelis Karkaletsis, Georgios Paliouras, Anastasia Krithara, and Elias Zavitsanos. Ontology population and enrichment: State of the art. *Knowledge-driven multimedia information extraction and ontology evolution*, pages 134–166, 2011.

Matthew E. Peters, Waleed Ammar, Chandra Bhagavatula, and Russell Power. Semi-supervised sequence tagging with bidirectional language models. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1756–1765, Vancouver, Canada, July 2017. Association for Computational Linguistics.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana, June 2018. Association for Computational Linguistics.

Bodo Plachta. *Editionswissenschaft: eine Einführung in Methode und Praxis der Edition neuerer Texte*. Number 17603. Reclam, 1997.

Borislav Popov, Atanas Kiryakov, Angel Kirilov, Dimitar Manov, Damyan Ognyanoff, and Miroslav Goranov. Kim–semantic annotation platform. In *International Semantic Web Conference*, pages 834–849. Springer, 2003.

Valentina Presutti, Francesco Draicchio, and Aldo Gangemi. Knowledge extraction based on discourse representation theory and linguistic frames. In *International Conference on Knowledge Engineering and Knowledge Management*, pages 114–129. Springer, 2012.

Jonathan K Pritchard, Matthew Stephens, and Peter Donnelly. Inference of population structure using multilocus genotype data. volume 155, pages 945–959. Genetics Soc America, 2000.

Michal Rosen-Zvi, Thomas L. Griffiths, Mark Steyvers, and Padhraic Smyth. The author-topic model for authors and documents. In *UAI '04, Proc. of the 20th Conference in Uncertainty in Artificial Intelligence, Banff, Canada, July 7-11, 2004*, pages 487–494, 2004.

Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall Press, USA, 3rd edition, 2009.

Marta Sabou, Kalina Bontcheva, Leon Derczynski, and Arno Scharl. Corpus annotation through crowdsourcing: Towards best practice guidelines. In *LREC*, pages 859–866, 2014.

Simon Schiff, Felix Kuhr, Sylvia Melzer, and Ralf Möller. Ai-based companion services for humanities. In *AI methods for digital heritage, Workshop at 43rd German Conference on Artificial Intelligence*, 2020.

Wei Shen, Jianyong Wang, and Jiawei Han. Entity linking with a knowledge base: Issues, techniques, and solutions. *IEEE Transactions on Knowledge and Data Engineering*, 27(2):443–460, 2015.

Pontus Stenetorp, Sampo Pyysalo, Goran Topić, Tomoko Ohta, Sophia Ananiadou, and Jun'ichi Tsujii. Brat: a web-based tool for nlp-assisted text annotation. In *Proceedings of the Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 102–107. Association for Computational Linguistics, 2012.

Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. Yago: a core of semantic knowledge. In *Proceedings of the 16th international conference on World Wide Web*, pages 697–706. ACM, 2007.

Hassina Nacer Talantikite, Djamil Aissani, and Nacer Boudjlida. Semantic annotations for web services discovery and composition. *Computer Standards & Interfaces*, 31(6):1108–1117, 2009.

Yee Whye Teh, Michael I Jordan, Matthew J Beal, and David M Blei. Hierarchical dirichlet processes. *Journal of the american statistical association*, 101(476):1566–1581, 2006.

Andrew Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE transactions on Information Theory*, 13(2):260–269, 1967.

Hanna M. Wallach. Topic modeling: beyond bag-of-words. In *Machine Learning, Proceedings of the Twenty-Third International Conference (ICML 2006), Pittsburgh, Pennsylvania, USA, June 25-29, 2006*, pages 977–984, 2006.

Chong Wang, David M. Blei, and David Heckerman. Continuous time dynamic topic models. In *UAI 2008, Proc. of the 24th Conference in Uncertainty in Artificial Intelligence, Helsinki, Finland, July 9-12, 2008*, pages 579–586, 2008.

Eva Wilden. *A Critical Edition and an Annotated Translation of the Akanānūru (Part 1-Kalirriyānainirai), vol. I: Introduction, Invocation–50; vol. II: 51–120; vol. III: Old Commentary on Kalirriyānainirai KV–90; Word Index of Akanānūru KV–120.* École Française d'Extrême-Orient, 2018.

Jie Yang, Yue Zhang, Linwei Li, and Xingxuan Li. YEDDA: A lightweight collaborative text span annotation tool. In *Proceedings of ACL 2018, Melbourne, Australia, July 15-20, 2018, System Demonstrations*, pages 31–36, 2018.

Alexander Yates, Michael Cafarella, Michele Banko, Oren Etzioni, Matthew Broadhead, and Stephen Soderland. Textrunner: open information extraction on the web. In *Proceedings of Human Language Technologies: The Annual Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*, pages 25–26. Association for Computational Linguistics, 2007.

Mohamed Amir Yosef, Johannes Hoffart, Ilaria Bordino, Marc Spaniol, and Gerhard Weikum. Aida: An online tool for accurate disambiguation of named entities in text and tables. *Proceedings of the VLDB Endowment*, 4(12):1450–1453, 2011.

Ce Zhang. *DeepDive: a data management system for automatic knowledge base construction.* PhD thesis, The University of Wisconsin-Madison, 2015.

Publications

The following workshop contributions, conference papers and journal articles have been peer-reviewed, accepted and published as part of this thesis.

Conference Papers

Felix Kuhr, Bjarne Witten, and Ralf Möller. Corpus-driven annotation enrichment. In *2019 IEEE 13th International Conference on Semantic Computing (ICSC)*, pages 138–141. IEEE, 2019

Felix Kuhr, Tanya Braun, Magnus Bender, and Ralf Möller. To Extend or not to Extend? Context-specific Corpus Enrichment. In *Proceedings of AI 2019: Advances in Artificial Intelligence*, pages 357–368. Springer, 2019

Felix Kuhr and Ralf Möller. Constructing and maintaining corpus-driven annotations. In *2019 IEEE 13th International Conference on Semantic Computing (ICSC)*, pages 462–467, Jan 2019

Felix Kuhr, Tanya Braun, and Ralf Möller. Augmenting and automating corpus enrichment. In *2020 IEEE 14th International Conference on Semantic Computing (ICSC)*, pages 69–76. IEEE, 2020

Felix Kuhr and Özgür L Özcep. Theory interpretations for topic models. In *The Thirty-Third International Flairs Conference*, 2020

Felix Kuhr, Magnus Bender, Tanya Braun, and Ralf Möller. Context-specific adaptation of subjective content descriptions. In *Proceedings of AI 2020: Advances in Artificial Intelligence*. Springer, 2020

Felix Kuhr, Magnus Bender, Tanya Braun, and Ralf Möller. Maintaining topic models for growing corpora. In *IEEE 14th International Conference on Semantic Computing, ICSC 2020, San Diego, CA, USA, February 3-5, 2020*, pages 451–458, 2020

Magnus Bender, Tanya Braun, Marcel Gehrke, Felix Kuhr, Ralf Möller, and Simon Schiff. Identifying subjective content descriptions among text. In *IEEE 15th International Conference on Semantic Computing, ICSC 2021, Virtual, January 27-29, 2021*, pages 451–458, 2021

Felix Kuhr, Magnus Bender, Tanya Braun, and Ralf Möller. Context-specific adaptation of subjective content descriptions. In *IEEE 15th International Conference on Semantic Computing, ICSC 2021, Virtual, January 27-29, 2021*, pages 451–458, 2021

Felix Kuhr, Matthis Lichtenberger, Tanya Braun, and Ralf Möller. Enhancing relational topic models with named entity induced links. In *IEEE 15th International Conference on Semantic Computing, ICSC 2021, Virtual, January 27-29, 2021*, pages 451–458, 2021

Journal Articles

Felix Kuhr, Tanya Braun, Magnus Bender, and Ralf Möller. Augmenting and automating corpus enrichment. *Int. J. Semantic Comput.*, 14(2):173–197, 2020

Magnus Bender, Tanya Braun, Marcel Gehrke, Felix Kuhr, Ralf Möller, and Simon Schiff. Identifying subjective content descriptions among text. *will be published in: Int. J. Semantic Comput.*, 15(4), 2021

Workshop Contributions

Tanya Braun, Felix Kuhr, and Ralf Möller. Unsupervised text annotations. In *Formal and Cognitive Reasoning - Workshop at the 40th Annual German Conference on AI (KI-2017)*, 2017

Simon Schiff, Felix Kuhr, Sylvia Melzer, and Ralf Möller. Ai-based companion services for humanities. In *AI methods for digital heritage, Workshop at 43rd German Conference on Artificial Intelligence*, 2020

Curriculum Vitae

Personal Information

Name	Felix Kuhr
Birthday	Juni 04, 1989
Place of birth	Hamburg, Germany
Nationality	German

Professional Experience

10/2016 - 08/2021	Research Assistant at the Institute of Information Systems, University of Lübeck
Since 10/2014	Founder and CEO of RedIP GmbH with a focus on communication networks
02/2011 - 10/2011	Network engineer at HTSM GmbH

Education

10/2011 - 04/2016	Hamburg University of Technology: Computational Informatics, Degree: M.Sc.
08/2008 - 02/2011	Büchner Kommunikationsnetzwerke GmbH: Apprenticeship, Degree: IT-Systemelektroniker
2006 - 2008	Lise-Meitner-Gymnasium, Hamburg
1999 - 2006	Gymnasium Othmarschen, Hamburg