Universität Zürich[UZH]

Master's thesis
presented to the Faculty of Arts and Social Sciences
of the University of Zurich
for the degree of
Master of Arts UZH

Automatic Labeling of Articles in International Investment Agreements

Using Semi-Supervised Learning and Word Embeddings

Author: Xi Rao
Student ID Nr.: 09-934-977

Examiner: Prof. Dr. Martin Volk

Supervisor: Dr. Kyoko Sugisaki

Institute of Computational Linguistics

Submission date: July 3, 2017

Abstract

International investment agreements (IIAs) are international commitments amongst contracting parties to protect and promote investment. Although each treaty has a distinctive structure in terms of placement and organization of information, IIAs as instruments of international law share underlying textual and legal structures. Treaty articles are important components in IIAs, with some articles titled and others untitled. In order to understand and analyze the treaty structure thoroughly, assignment of titles to each article is crucial for content analysis. In this master thesis, we attempt to automatically assign titles to untitled international investment treaty articles using semi-supervised learning. Various titles have been assigned to similar texts due to the variability of negotiating partners, languages, traditions, etc. Hence, in order to have a condensed representation of various article titles, we firstly cluster 34,524 titled articles into ten topics by expanding word and document semantics with embeddings. We then use these ten classes as the labels in our classification task where titles are assigned to 10,074 untitled articles. The classification task is performed with supervised classifiers (k-nearest neighbors (KNN), support vector machine (SVM), multi-layer perceptron (MLP), stochastic gradient descent (SGD), convolutional neural network (CNN)) and partially supervised k-means clustering. Expert annotations of 100 untitled articles are used as the gold standard in our evaluation set. K-means clustering with the retrained word embeddings tailored to our corpus has brought about an increase of 30% in accuracy compared to a simple CNN classifier, which has scored the highest amongst all supervised classifiers. The comparison between these two machine learning paradigms (supervised and semi-supervised learning) leads to the conclusion that word embeddings can effectively expand the semantic features for words and documents, which allows us to perform accurate categorization of texts from closely related sub-fields of one research area, for instance, to categorize the ten topics in the study of IIAs.

Zusammenfassung

Internationale Investitionsabkommen (IIAs) sind internationale Verträge, die dem Schutz und der Förderung von Investitionen dienen. Obwohl jedes IIA einen vertragsspezifischen Aufbau aufweist, liegt ihnen eine gemeinsame inhaltliche und rechtliche Struktur zu Grunde. Vertragsartikel sind ein integraler Bestandteil von IIAs, deren Titel allerdings nicht immer explizit benannt werden. Für eine detaillierte Analyse der Verträge und ihrer Struktur ist eine vollständige Betitelung der Artikel allerdings unerlässlich. Ziel dieser Masterarbeit ist es, Artikeln ohne Titeln diese mit Hilfe von semi-supervisiertem Lernen zuzuordnen.

Eine Schwierigkeit liegt darin, dass inhaltlich ähnlichen Textabschnitten häufig keine identischen Titel gegeben werden. Dies liegt an der Fülle an Verhandlungspartnern, Sprachen und Konventionen, die an der Erstellung dieser Verträge beteiligt sind. Um eine Liste repräsentativer Titel zu erhalten, verwenden wir Clustering, um zunächst 34.524 Artikel mit Titel zehn verschiedenen Themenbereichen zuzuordnen. Dabei erweitern wir Wort- und Dokumentsemantik mit Word und Document Embeddings. Wir verwenden diese zehn Kategorien als Labels, um 10.074 unbetitelten Artikeln Titel hinzuzufügen. Wir führen diese Klassifizierung mit Hilfe von supervisierten Klassifikatoren (k-Nearest Neighbor (KNN), Support Vector Machines (SVM), Multilayer Perceptron (MLP), Stochastic Gradient Descent (SGD), Convolutional Neural Networks (CNN)) und dem semi-supervisierten k-Means Clustering-Algorithmus. Die Expertenannotierung von 100 Artikeln ohne Titel dient als Goldstandard für die Evaluierung.

Wir stellen fest, dass CNN der beste Klassifikator für unser Lernproblem ist und dass die Verwendung des k-Means Clustering-Algorithmus mit vortrainierten, auf unseren Korpus angepassten Word Embeddings im Vergleich zu einem einfachen CNN-Klassifikator zu einer Verbesserung der Accuracy um 30% führt. Wir schliessen aus diesem Vergleich zwischen zwei Paradigmen des maschinellen Lernens (supervisiertes und semi-supervisiertes), dass Word Embeddings die semantischen Merkmale von Wörtern und Dokumenten erfolgreich erweitern können. Dies ermöglicht uns eine genaue Kategorisierung von Texten, die aus eng verwandten Teilgebieten eines Forschungsfeldes stammen, so wie zum Beispiel die Kategorisierung von zehn Themenbereichen innerhalb der IIAs.

Acknowledgement

First and foremost, I would first like to thank my thesis advisor Prof. Dr. Martin Volk of the Institute of Computational Linguistics at the University of Zurich. The door to Martin's office was always open whenever I ran into a trouble spot or had a question about my research, writing or coding. He has been offering great support during the two years of my master's program in Multilingual Text Analysis (major) and Computational Linguistics (minor), whenever help and advice are needed. I have been extremely lucky to have an advisor who cared so much about my work, and who responded to my questions and queries so promptly.

I would also like to thank the mentors who were involved in the project for brainstorming, idea sketching and annotation: Dr. Kyoko Sugisaki from the Institute of Computational Linguistics, University of Zurich; Prof. Dr. Peter Egger from KOF Swiss Economic Institute, ETH Zurich. Without their passionate participation and input, this thesis could not have been successfully conducted. I thank Kyoko for providing me with corpus materials and answering those lengthy emails I sent for advice. I am also grateful to have Peter sitting in front of his computer for hours and annotating the treaty articles for us; the evaluation would have been impossible without his great efforts and expertise.

I would also like to thank Sophia Ding, Mathias Müller and Dr. Kyoko Sugisaki for proofreading my thesis, and providing very valuable comments. I am grateful that Sophia and Mathias, two German native speakers, helped me reformulate the German version of abstract. All remaining errors are my own.

The Institute of Computational Linguistics has provided me with the support and equipment to produce and complete my thesis. During my master studies, the institute and faculty have been very supportive and friendly. Hereby I would love to thank Laura Mascarell and Dr. Annette Rios for providing me with in-house translation of Spanish and French treaties. I am greatly indebted to Jeannette Roth and Dr. Manfred Klenner for their assistance, coordination and consultation hours during my master studies. Thanks go to Dr. Simon Clematide for the lectures on machine learning and deep learning as well as helping me (together with Samuel

Läubli, thank you) make peace with our GPU server *rattle*. Andrea Fritz and Yvonne Gwerder, thank you for spending those valuable discussion time with me during my entire master studies.

I would love to thank the SNIS project team of *Diffusion of International Law* for letting me use their preprocessed XML documents. I am also indebted to the many countless contributors to the open-source programming community for providing the numerous tools and libraries I have used to produce both my results and this thesis.

Finally, I must express my very profound gratitude to my parents, parents-in-law and my husband for providing me with unfailing support and continuous encouragement throughout my years of studies and through the process of researching and writing this thesis. This accomplishment would not have been possible without them. Thank you. I will be grateful forever for your love.

The main insight learned from interdisciplinary studies is the return to specialization.

by George Stigler

Contents

List of Figures

List of Tables

List of Listings

List of Abbreviations

ACL	Association for Computational Linguistics
AP	Affinity Propagation
AUC	Area Under the Curve
BITs	Bilateral Investment Treaties
BoW	Bag-of-Words
CBOW	Continuous Bag-of-Words
CL	Computational Linguistics
CNB	Complement Naive Bayes
CNN	Convolutional Neural Network
DBOW	Distributed Bag-of-Words
DMPV	Distributed Memory of Paragraph Vector
IIAs	International Investment Agreements
KNN	K-nearest Neighbor
LDA	Latent Dirichlet Allocation
MDS	Multidimensional Scaling
MLP	Multi-layer Perceptron
MNF	Most Favored Nation
MT	Machine Translation
NB	Naive Bayes
NLP	Natural Language Processing
NLTK	Natural Language Toolkit
NMF	Non-negative Matrix Factorization
OCR	Optical Character Recognition
OvO	One-vs-one
OvR	One-vs-the-rest
PDF	Portable Document Format
PoS	Part-of-Speech
ReLU	Rectified Linear Unit

ROC	Receiver Operator Characteristics
SGD	Stochastic Gradient Descent
SMT	Statistical Machine Translation
SNIS	Swiss Network for International Studies
SVD	Singular Value Decomposition
SVM	Support Vector Machine
TF/IDF	Term Frequency/Inverse Document Frequency
TIPs	Treaties with Investment Provisions
UK	United Kingdom
UNCTAD	United Nations Conference Trade and Development
USA	United States of America
XML	eXtensible Markup Language

1 Introduction

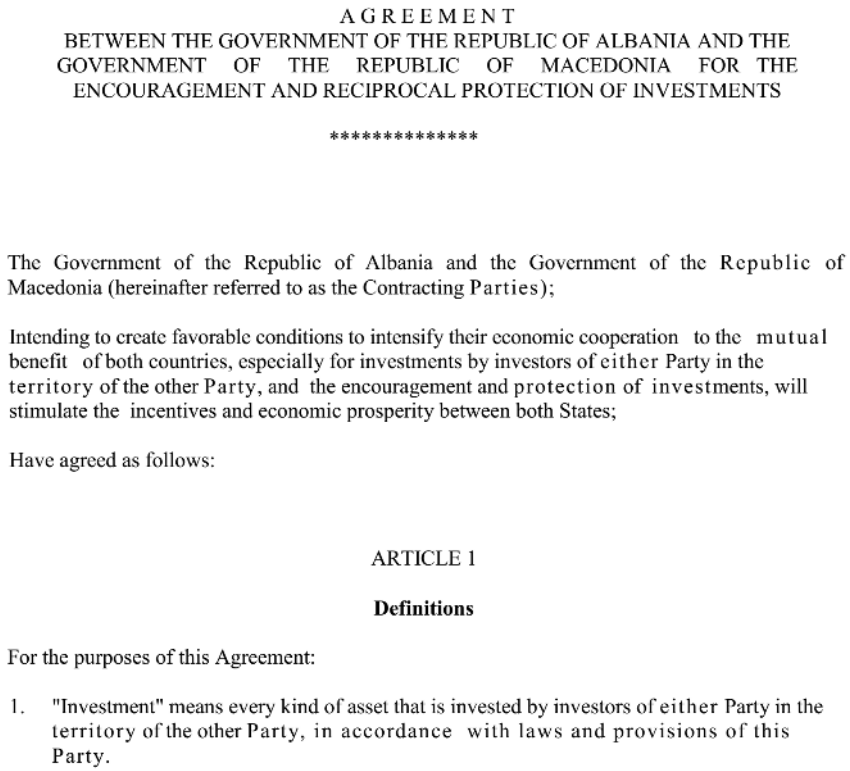
International investment agreements (IIAs) are "essentially instruments of international law" [Salacuse, 2015, 1]. A fundamental purpose of investment treaties is to protect and promote investment. Contracting parties "make commitments with respect to the treatment they will accord to investors and investment from those other parties, and agree to some mechanism for enforcement of those commitments" [ibid.].

IIAs can be divided into three types: (1) bilateral investment treaties (BITs), (2) treaties with investment provisions (TIPs) and (3) other investment-related agreements involving more than two contracting parties[1]. Although BITs account for the great majority of IIAs, the provisions of IIAs can vary greatly from one to another due to the scope of negotiation.

It has been commonly agreed on that in the literature for treaty content and structure, although there has been no uniform treaty structure and the degree of agreement varies across treaties, essentially all investment treaties address the same issues and follow similar legal and textual structures (see Salacuse [2015]; Alschner and Skougarevskiy [2016a]). As a result, despite the variations in language usage from treaty to treaty, we argue that because of the strong commonality among them, more than 3,300 individual investment treaties negotiated over the last six decades constitute a single, integrated global regime for investment. In the field of IIAs, the term *regime* is generally understood to consist of four elements (1) principles, (2) norms, (3) rules and (4) decision-making process [Salacuse, 2015, 10].

[1] Summarized based on Salacuse [2015, 1] and information offered by United Nations Conference Trade and Development (UNCTAD). UNCTAD offers an extensive overview on terminologies of IIAs. A BIT is an agreement between two contracting parties "regarding promotion and protection of investments made by investors from respective countries in each other's territory". "TIPs bring together various types of investment treaties that are not BITs". UNCTAD defines TIPs in three subtypes: "broad economic treaties that include obligations commonly found in BITs (e.g. a free trade agreement with an investment chapter); treaties with limited investment-related provisions (e.g. only those concerning establishment of investments or free transfer of investment-related funds); and treaties that only contain "framework" clauses such as the ones on cooperation in the area of investment and/or for a mandate for future negotiations on investment issues". More detailed explanations can be found on http://investmentpolicyhub.unctad.org/IIA (accessed 20 May 2017).

Generally speaking, a treaty is composed of preface, preamble (e.g. title page and table of contents), text body (i.e. articles and paragraphs), conclusion (e.g. signatures) and sometimes annex [Sugisaki et al., 2016, 205], with articles as thematic units. Figure 1 shows an example of a preamble and a titled article (Article 1 entitled "Definitions") in an IIA. In Figure 2 we provide an example of an untitled article.



AGREEMENT
BETWEEN THE GOVERNMENT OF THE REPUBLIC OF ALBANIA AND THE GOVERNMENT OF THE REPUBLIC OF MACEDONIA FOR THE ENCOURAGEMENT AND RECIPROCAL PROTECTION OF INVESTMENTS

****

The Government of the Republic of Albania and the Government of the Republic of Macedonia (hereinafter referred to as the Contracting Parties);

Intending to create favorable conditions to intensify their economic cooperation to the mutual benefit of both countries, especially for investments by investors of either Party in the territory of the other Party, and the encouragement and protection of investments, will stimulate the incentives and economic prosperity between both States;

Have agreed as follows:

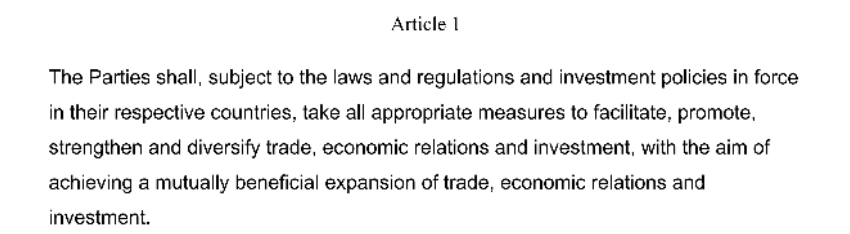
ARTICLE 1

Definitions

For the purposes of this Agreement:

1. "Investment" means every kind of asset that is invested by investors of either Party in the territory of the other Party, in accordance with laws and provisions of this Party.
In particular, though not exclusively, the term "Investment"means:

Figure 1: Example of a preamble and a titled article



Article 1

The Parties shall, subject to the laws and regulations and investment policies in force in their respective countries, take all appropriate measures to facilitate, promote, strengthen and diversify trade, economic relations and investment, with the aim of achieving a mutually beneficial expansion of trade, economic relations and investment.

Figure 2: Example of an untitled article

In order to understand the negotiation behaviors of contracting parties, we can use the content and structure of IIAs as a posteriori proxies and reflection of the negotiation processes. Therefore, analyzing treaty structure and content of IIAs as

a body of law instruments has established itself as a research area that continuously gains more interest from various disciplines, such as law, economics, political science.

1.1 Motivation

A joint project on IIAs was launched from various disciplines (law, economics, political science, computational linguistics (CL)) under the Swiss Network for International Studies (SNIS) network for a project called *Diffusion of International Law: A Textual Analysis of International Investment Agreements*[2] with the goals to understand the design, evolution, and effects of the IIAs currently in practice.

The project is still ongoing, one of whose goals is to create a complete, up-to-date text collection of IIAs, including various types of texts, in one single format and standardized by language (see Sugisaki et al. [2016] for the current status of the corpus). Another aim of the SNIS project is to create a database on IIAs based on the collection of treaties, which will then serve as the "empirical backbone for answering a set of important questions related to understanding the design, evolution and effects of IIAs". This database is of further use to "provide new measures for structures and the content of treaty texts", to explore textual similarities across treaties, to extract "the patterns of diffusion and to link different measures of treaty design with outcomes such as investment flows"[3]. Subsequently, the SNIS project aims at developing a new toolkit for treaty negotiations as well as arbitration.

The master thesis has been motivated by various inputs from an interdisciplinary team. First, it started with a small task Dr. Kyoko Sugisaki gave me during my master studies. The task was to separate the text segments of English from Chinese in a *Microsoft Word* document of a bilingual BIT. This is how I was acquainted with the SNIS project on IIAs. Then at KOF Swiss Economic Institute where I currently work, I learned from Prof. Dr. Peter Egger that he has launched an IIA coding project where information extraction and classification is of interest to content and structure analysis in IIAs. After a short discussion with Prof. Dr. Martin Volk, we were all fascinated by the idea of classifying IIA treaty articles which can then lead to useful NLP applications such as information retrieval and extraction across languages. We believe that treaty article categorization can assist mapping treaty texts to their inherent structures. The resulting simplified structure of a treaty is

[2]http://www.snis.ch/project_diffusion-international-law-textual-analysis-international-investment-agreements (accessed 26 Jan 2017).

[3]See the project description at http://www.snis.ch/project_diffusion-international-law-textual-analysis-international-investment-agreements (accessed 26 Jan 2017).

represented by certain categories of articles, which is in turn beneficial to organize treaties in information retrieval systems or databases.

Currently, to the best of our knowledge, United Nations Conference Trade and Development (UNCTAD) offers the only query system for IIAs. It provides its users with an online *IIA Navigator*[4] where the user is allowed to perform basic metadata queries for IIAs, such as contracting party, contracting year, region, type of agreement, status, text availability, relation with other treaties. UNCTAD also offers another database, *IIA Mapping* Project[5], where treaties can be queried for certain elements such as "standards of treatment". The mapping database acts as a tool "to understand trends in IIA drafting, assess the prevalence of different policy approaches and identify treaty examples"[6]. As far as we are concerned, the database was created based on human annotations, e.g. without automatic efforts from machinery. The query results are a list of treaties described by values customized to the users' input (e.g. type of most-favored-nation (MFN) clause: post-establishment). Original texts of treaties are in most of the cases provided as PDF documents in the original language(s) of publication and queries of full-text are impossible with scanned PDF documents.

As meaningful subunits in an IIA treaty, treaty articles are smaller units to understand the treaty content and structure, because treaty articles are composed of sentences that are formulated coherently to convey meaning expressed in one article. That being said, as the first step for a more fine-grained database of IIAs, we would like to categorize treaty content in a structured manner, e.g. by categorizing treaty articles. In view of this thesis, we can build up a comprehensive database in the future with categorized articles, where a full-text query of specific types of provisions is allowed.

1.2 Research Question

Current research on text classification in the legal domain has mainly focused on a document as a whole. Treaty article as the unit of analysis has not yet been the focus of research in the legal domain, although the exploration of smaller analysis unit (e.g. sentential, see Bartolini et al. [2004]; de Maat and Winkels [2008, 2009, 2010]) has begun. When we look at the IIA treaties at the article level, some articles come with titles; others do not. As most of the treaty articles are marked with titles

[4]http://investmentpolicyhub.unctad.org/IIA (accessed 10 Jan 2017).

[5]http://investmentpolicyhub.unctad.org/IIA/mappedContent (accessed 10 Jan 2017).

[6]http://investmentpolicyhub.unctad.org/IIA/mappedContent (accessed 10 Jan 2017).

which summarize the content described succinctly, we can utilize article titles as an assistance to grasp the structure and content in treaties without reading through treaty texts. Hence, if we aim at representing treaty structure with articles, we will first need to assign titles to the untitled text blocks by learning the knowledge encoded in the titled articles.

The research question that shall be answered in this thesis is how to apply text categorization methods developed in the community of Natural Language Processing (NLP) and CL to assign article titles to untitled treaty articles automatically.

Concretely speaking, we will investigate the applicability of machine learning methods (e.g. supervised, unsupervised and semi-supervised) to assign titles to untitled articles, where the efficacy of different methods will be evaluated by their accuracy of assigning the correct titles. Last but not least, to better understand the quality of machine-generated titles, agreement tests between human-chosen titles and machine-generated titles will be conducted.

1.3 Thesis Structure

This thesis is organized into seven chapters. In this first chapter, we have introduced the motivation and research question. Moreover, we outline the structure and contributions of the thesis. Chapter 2 provides a literature review on the endeavors of using IIA treaty texts as data in various disciplines, as well as important literature from NLP and CL on text similarity and its applicability to text categorization using different machine learning methods. Our corpus and preprocessing steps are presented in Chapter 3. Chapter 4 describes our pipeline for text categorization and the specific setups we adopt for the experiments, followed by Chapter 5, an extensive result analysis and evaluation. We conclude the paper with Chapter 6 about important findings and implications and Chapter 7 about future work.

1.4 Contributions of the Thesis

The contributions of this master thesis are as follows:

- This work bridges the literature between IIAs and CL by outlining the application of methods developed in CL and NLP in studying IIAs.
- An extensive literature review is presented on textual similarity, its subcate-

gories and their application in text categorization.

- This thesis discusses and tests various methods to generate document embeddings using pretrained and retrained word embeddings.
- Various techniques of machine learning applicable to text categorization have been tested and compared.
- To the best of our knowledge, this work is the first endeavor to categorize treaty articles in IIAs.

2 Literature Review on Text as Data: Textual Similarity and Text Categorization

Texts are ubiquitous existence in our written culture. Thus, textual analysis prevails in many disciplines due to the necessity of transforming texts into reasonable data of analysis, such as numbers. Various endeavors have been made from the different disciplines such as law, economics, to understand the structure of IIAs and map the content of treaties to some corresponding thematic topics. Only recently, textual analysis has become a popular method to analyze IIAs. How to deal with text as data and how to employ methods and tools from NLP and CL have emerged in the recent literature of studying IIAs (see Alschner and Skougarevskiy [2015, 2016a,b]). In this chapter, we review textual analysis from the perspectives of its applicability in IIA studies and discuss the useful techniques in NLP and CL, which can be employed to perform content and structure analysis in IIAs, i.e. textual similarity measurement and text categorization.

2.1 Treaty Texts as Data: Previous Work on Textual Analysis in International Investment Agreements (IIAs)

Text as data has become a central issue in understanding the structure and content of IIAs. In the recent literature of IIAs, there has been growing interest in measuring textual similarity (as a proxy for legal similarity) of treaties across countries and therewith comparing the characteristics of country negotiation patterns. The Jaccard distance (see Section 2.3.1) has been adopted in Alschner and Skougarevskiy [2015, 2016a,b] to compute dissimilarity between IIA treaties. They first split the treaty texts into character 5-grams, i.e. five consecutive characters. For example,

given a sentence "This is a sentence", the units after the split are "This_", "this_i", "is_is", ..., "tence" (white spaces marked by underscore "_"). Then they used the Jaccard distance to measure the dissimilarity of the split texts. Based on the computed text similarity for IIAs, the authors have conducted studies on the correlation between IIA textual similarity and economic bargaining power, economic development, rule making, policy consistency, and innovation, etc. Two interesting products of the literature are summarized at the website of the project *Mapping BITs*[1]:

1. Affinity propagation (AP)[2] clustering was performed using the dissimilarity matrix of treaties. Twenty closest neighbors in terms of Jaccard distance for each treaty were located and visualized with heat map[3].

2. The similar textual segments (aka character 5-grams) on the article levels were mapped between treaties to develop a better understanding of patterns underlying international economic law[4].

Another endeavor with text as data is to apply network analysis based on textual similarities for investigating "patterns of convergence and divergence in international trade and investment law"[5].

Techniques of textual analysis (e.g. clustering, text similarity measures) have been receiving much attention due to their efficacy in transforming textual data into meaningful and operationalizable representations. Some interesting attempts on the potential of applying textual analysis to IIAs have been carried out as we can see from this brief review in this section, yet there are still some critical issues to tackle:

- The unit of analysis (what counts as a document in a textual analysis) still remains at the level of the treaty. Although Alschner and Skougarevskiy [2016a] have briefly mentioned mapping the similar and dissimilar segments across treaties on the level of articles, there is little work on exploring articles, their titles and their inherent topics. Our aim is to determine whether treaties share a similar inherent structure; hence, it is crucial to conduct semantic analysis

[1] http://mappinginvestmenttreaties.com/ (accessed 26 Jan 2017).

[2] Sarkar [2016, 308] introduces AP as an algorithm that "tries to build clusters based on inherent properties of the data" without specifying the number of clusters in advance. See Section 2.5 for more details on clustering. The difference between k-means and AP clustering lies in the existence of assumption about the number of clusters.

[3] See *Methodology* section under http://mappinginvestmenttreaties.com/ (accessed 26 Jan 2017).

[4] One example shown in http://mappinginvestmenttreaties.com/specials/tpp/ (accessed 26 Jan 2017).

[5] http://graduateinstitute.ch/home/research/centresandprogrammes/ctei/projects/text-as-data-analysis-of-IEL.html (accessed 26 Jan 2017).

on the level of articles.

- Despite the interest in multilingualism of IIAs, there has not been a systematic study on the IIA mapping across languages. Alschner and Skougarevskiy [2016a] experimented with 1,628 English, 306 French and 165 Spanish treaties and only performed similarity measures and treaty clustering within the same language. Machine translation of Spanish and French treaties into English is the fastest method to obtain more English material to understand the IIA structure across languages.

Previous work on IIAs has mainly focused on the document level. To uncover a hidden structure of a treaty, we need to extend our unit of analysis to treaty article. The following sections of the literature review are devoted to the measurement of textual similarity in NLP, text categorization using machine learning techniques and the interplay of textual similarity measures and machine learning techniques.

2.2 Text Categorization: Machine Learning and Feature Engineering

Generally speaking, text categorization is a task in NLP, where a new document is assigned to "one of a pre-existing set of document classes" [Jurafsky and Martin, 2009, 844]. It is commonly agreed upon that supervised machine learning is a standard approach of text categorization [ibid., 844].

However, in a broader context of machine learning techniques, three types have been applied to text categorization, namely, supervised text classification, unsupervised text clustering, and semi-supervised text categorization. In this master thesis, text categorization is used as a hypernym of text classification and text clustering, as the latter two terms refer concretely to text categorization under supervised and unsupervised settings, respectively. However, in the literature, there seems to be no clear distinction made between the terms *classification* and *categorization*. The two terms are often used interchangeably, regardless of the settings of machine learning (see Sarkar [2016, 167] as an example).

Throughout this thesis, we make a clear distinction among the three terms. *Text classification* is defined as "trying to organize text documents into various categories based on inherent properties or attributes of each text document" [ibid., 167] with supervised learning techniques. *Text clustering* is also known as *document clustering*, where documents are clustered into groups "purely based on their features, similarity

and attributes, without training any model on previously labeled data" [ibid., 170]. The term *text categorization* is referred here as the hypernym of the previous two terms; therefore it is used in this thesis as the broadest term to address labeling texts with certain taxonomy.

A machine learning model has two interlinking parts, i.e. data and algorithm [ibid., 167]; hence, we first discuss the difference in data and algorithms among supervised, unsupervised and semi-supervised learning, respectively.

Supervised learning requires pre-labeled data samples, while unsupervised learning does not require any pre-labeled samples to build a model. Feature patterns from unlabeled data are learned by grouping together similar data points in an unsupervised learning, whereas feature sets are extracted from each labeled sample for each class in a supervised setting (see Raschka [2015, 3, 6], Sarkar [2016, 170]). Both supervised and unsupervised techniques allow us to make predictions about the group membership of unseen data (aka test data, holdout data). The biggest advantage of supervised learning is that it allows class-specific feature engineering that might increase the accuracy of predictions, as we are provided with labeled instances corresponding to the class labels. Quite the contrary, in a setting of unsupervised learning, we often deal with "unlabeled data or data of unknown structure" [Raschka, 2015, 3]; hence, techniques such as clustering or topic modeling enable representations of inherent data structure, as well as identification of group membership.

Semi-supervised techniques have been introduced as an intermediate solution to combine the advantages of supervised and unsupervised learning. A definition and typical settings for semi-supervised learning are provided by Sammut and Webb [2011, 897] from the perspective of text processing: A semi-supervised system "takes as input a (small) training set of labeled examples and a (larger) working set of unlabeled examples". They have also pointed out that in a semi-supervised learning, we usually evaluate a learner's performance "on a test set that consists of unlabeled examples".

What is commonly important among three types of machine learning techniques is feature engineering (aka feature extraction) which is defined as the process to extract and select features from our data [Sarkar, 2016, 178]. Within the context of text categorization, features are "unique, measurable attributes" [ibid., 177] for each text snippet in our corpus. They can be characters, words, or even phrases. In the literature, there are three types of popular techniques of feature engineering (i.e. vectorization, transformation from text tokens to numerical vectors [ibid., 221]): Bag-of-Words (BoW) model, Term Frequency/Inverse Document Frequency (TF/IDF) BoW model and word embedding (see ibid., 178-193).

BoW model is a simple yet powerful vector space model, where we represent each text snippet as a vector of vocabulary counts in the corpus. Vector space in the BoW model is defined by the corpus vocabulary as dimensions. Regarding the following two sentences in Example 2.1[6] (as a small corpus). We remove the stop word "the" and the punctuation "." to generate Example 2.2. The vocabulary of *s1* and *s2* is "cat", "mouse", "ate", "food" which are dimensions in the vector space. Based on the document term matrix in Table 1, we can obtain the vector representations of $s1 = [1, 1, 1, 0], s2 = [1, 1, 1, 1]$.

(2.1) The cat ate the mouse.
The mouse ate the cat food.

(2.2) *s1* = ["cat","ate","mouse"]
s2 = ["mouse","ate","cat","food"]

		dimensions			
		cat	ate	mouse	food
documents	*s1*	1	1	1	0
	s2	1	1	1	1

Table 1: Document term matrix for Example 2.2 with the BoW model

The disadvantage of the BoW model lies in the usage of the absolute frequency of words in documents. It does not consider the relative importance of a word in relation to each document. As a result, TF/IDF model of feature extraction has been introduced where we multiply the term frequency (TF) and inverse document frequency (IDF) metrics. Add-one smoothing is used to prevent potential division-by-zero error [Sarkar, 2016, 182]. Given N as the total number of documents in corpus, t as a term, log with base e, $df(t)$ as the number of document where the term t appears, we calculate the IDF by $1+log\frac{N}{1+df(t)}$ [ibid., 182]. Taking the absolute term frequency shown in Table 1, we compute the vectors for the term "cat" in *s1* and *s2* with the TF/IDF-weighted BoW model.

$$cat_{s1} = cat_{s2} = 1 \times (1 + ln\frac{2}{1+2}) = 0.595$$

[6]The examples are taken from an inspiring blog on text similarity, see http://text-analytics101.rxnlp.com/2015/11/understanding-text-similarity.html (accessed 25 May 2017).

		dimensions			
		cat	ate	mouse	food
documents	*s1*	0.595	0.595	0.595	1
	s2	0.595	0.595	0.595	0.595

Table 2: Document term matrix for Example 2.2 with the TF/IDF-weighted BoW model

We can see from Table 1 and Table 2 that the vectors computed for the same document are different in numbers even if the vector space has not changed. More discussion on the consequence of various techniques of vectorization can be found in Section 2.3. Another more advanced technique to transform documents into vector representation is word embedding for which we provide an extensive review in Section 2.3.3.

Through the above discussion of techniques in feature extraction concerning textual data, we can conclude that methods of feature engineering can influence the results of machine learning tremendously regardless of algorithm. For instance, we can turn Example 2.2 into a simple learning problem by asking the question: Do the two sentences denote the same meaning? Apparently, we human can comprehend that the two sentences are entirely different in meaning. For computers, to distinguish their meaning, it is required to have a priori knowledge of semantics and syntax, because for instance, the word "cat" in *s1* and that in *s2* do not bear the same syntactic function. Even from this simple example, we can see that clear differentiation of various types among text similarities has an enormous impact on the techniques we choose in NLP tasks as well as the evaluation of task performance.

Unfortunately, the degree of attention of the interconnectivity between textual similarity and machine learning techniques varies across different types of techniques. It fails to explain the relationship between supervised learning methods and textual similarity, while textual similarity and unsupervised learning methods such as clustering is more often the topic in the literature. Essentially, the applicability of machine learning techniques onto textual categorization is to learn the textual cohesion and coherence[7] from the features that encode either textual similarities or

[7]Fasold and Connor-Linton [2014, 511] define *coherence* as "the overall sense of a discourse that results from relationships (a) within a sequence of utterances and (b) between those utterances and their context"; they define *cohesion* as "a sense of unity within a text that results from language that connects a current point in the text to a prior part of a text".

dissimilarities (e.g. see discussion in Bird et al. [2009, 236]). The more similar two texts are, the more likely they belong to the same category. In the upcoming section, we briefly introduce three different types of textual similarity and their most popular measures: surface lexical similarity (Section 2.3.1), distributional semantic similarity (Section 2.3.2) and word embedding similarity (Section 2.3.3). The three types of similarities do not necessarily entail each other as shown in Example 2.2. We will see from the literature review of machine learning (Sections 2.4, 2.5 and 2.6) that all techniques of text categorization can be linked to compute the (dis)similarity of texts regardless of the granularity of analysis (e.g. character, token, phrase, sentence).

2.3 Text Categorization and Textual Similarity

The distinction between two types of text similarity has been clarified in the *SemEval-2016 Task 1* [Agirre et al., 2016, 500]: surface lexical similarity and word embedding similarity. Surface lexical similarity originated from the "information theoretic measure based on unigram overlap" [ibid., 500], where only the surface difference of strings is compared, either at the granularity level of character level, word or phrase. Regarding Example 2.2, the surface lexical similarity of two sentences is high because they share the common overlapping content words "cat", "ate", and "mouse", with stop word "the" and punctuation "." removed.

On the other hand, word embedding similarity is often understood as distributional similarity [Jurafsky and Martin, 2009, 693], as both of the concepts build on the famous statement of Firth [1957, 11]: "You shall know a word by the company it keeps". Word embedding was derived from the community of deep learning, which was not influenced by the word-counting distributional paradigm developed in CL [Baroni et al., 2014, 239]. In this thesis, we distinguish between distributional semantic similarity and word embedding similarity, following the typology defined in Baroni et al. [ibid.] of context-counting distributional semantics and context-predicting word embedding. Both distributional and word embedding similarity account for measurement of meaning representation, with the former focused on counting context, the latter predicting context.

2.3.1 Surface Lexical Similarity

From Example 2.2, we see that surface similarity does not involve further interpretation of the meaning of words. It simply measures the textual similarity between the representations of surface forms. Two similarity measures on surface lexical

similarity are introduced here: Levenshtein and Jaccard distance.

Levenshtein distance The Levenshtein distance, also known as edit distance, measures the string difference of sequences. This metric computes the minimum number of single-character edits (insertions, deletions or substitutions) between two words[8], if we change from one word to the other. Each operation has a cost (usually set to 1). The edit distance of words can be extended to edit distance of phrase, where we count the minimum number of operations to change from one phrase to another with words as the basic unit. In Example 2.2, the edit distance between two phrases is calculated as the minimum steps of moving from *s1* to *s2* in Example 2.2. As word order matters in edit distance, we have one substitution of "mouse" for "cat", another substitution of "cat" for "mouse", and an insertion of "food" when changing from *s1* to *s2*. As a result, the Levenshtein distance between *s1* and *s2* is 3. Since the calculation of edit distance is symmetric between two phrases, changing from *s2* to *s1* has the same number of operations (3, two substitutions and one deletion).

The Levenshtein distance can be normalized, so that the results of edit distances is comparable with other similarity measures[9]. Two possibilities for edit distance in normalization: (1) take the norm of the longer sequence; (2) take the norm of the shorter sequence. Norm is the length of a sequence. In Example 2.2, the normalized edit distance using the norm of shorter sequence (*s1*) is $\frac{3}{3} = 1$, while that using the norm of longer sequence (*s2*) is $\frac{3}{4} = 0.75$.

Jaccard distance Another string similarity measure is the Jaccard coefficient. It was originally designed for binary vectors and extended later to vectors of weighted associations (Jurafsky and Martin [2009, 699])[10].

$$sim_{Jaccard}(\vec{v}, \vec{w}) = \frac{\sum_{i=1}^{N} min(\vec{v}, \vec{w})}{\sum_{i=1}^{N} max(\vec{v}, \vec{w})}$$

$\vec{v}$ and $\vec{w}$ denote the vector representations of two sequences. This metric computes the intersection of two sequences (in terms of their identical elements) divided by all possible elements in $\vec{v}$ and $\vec{w}$. Subtracting the Jaccard coefficient from 1, we obtain the Jaccard distance which can be understood as a

[8] https://en.wikipedia.org/wiki/Levenshtein_distance (accessed 01 May 2017).

[9] https://pypi.python.org/pypi/Distance/ (accessed 02 May 2017).

[10] Mathematical formulas in Section 2.3.1 and Section 2.3.2 are taken from Jurafsky and Martin [2009, Chapter 20: Computational Lexical Semantics, 697-699] and the notations are slightly adapted.

dissimilarity measure[11]:

$$1 - \frac{intersection}{union}$$

For *s1* and *s2* in Example 2.2, the intersection of *s1* and *s2* is 3 (words "cat", "mouse", "ate"), the union is 4 (the vocabulary of *s1* and *s2*, "cat", "mouse", "ate", "food"). Thus, the Jaccard distance is $1 - \frac{3}{4} = 0.25$.

To summarize, surface similarity measures the similarity of surface forms and does not incorporate the semantics encoded in words and sentences.

2.3.2 Distributional Semantic Similarity

Distributional semantics can be described in a nutshell as: We compute vector representations for each word by counting co-occurrences in the word's context in large corpora. Then we perform dimensionality reduction to reduce the sparsity in the vector representations. Optimization of the vector representations can be adjusted with tuning parameters such as context window, association coefficient, and vector dimensionality techniques (see Baroni et al. [2014]).

Jurafsky and Martin [2009, 693] summarize nicely three parameters to consider when computing distributional similarity measures: (1) co-occurrence (i.e. what count as neighbors); (2) how are co-occurrences weighted (e.g. binary, frequency or mutual information[12]); (3) vector distance measures (e.g. cosine similarity, Euclidean distance).

Regarding co-occurrence [ibid., 693-694], one can either look for the neighboring items in plain text or syntactic relations, with stop words filtered and the context windows ranging from ± 1 to ± 500. The weights for features can be binary (indicating whether items co-occur), the absolute or relative frequency of the words [ibid., 695-697].

Using our Example 2.2, to compute distributional similarity measures, we first need to construct a document term matrix. For documents *s1* and *s2*, we count the frequency of words in the vocabulary in the documents. This is the calculation we have performed in Section 2.2 when explaining the BoW model, from which we obtain the vector representations of $s1 = [1, 1, 1, 0]$ and $s2 = [1, 1, 1, 1]$.

In distributional semantics, every word is represented by a vector. To define simi-

[11] http://www.wow.com/wiki/Jaccard_index (accessed 20 May 2017).

[12] Mutual information calculates how often two words co-occur compared with what we would expect if they were independent [Jurafsky and Martin, 2009, 696].

larities of two words, we essentially calculate the similarity of two vectors. In this section, three important metrics are introduced: Manhattan distance (aka L1 norm), Euclidean distance (aka L2 norm) and cosine similarity.

Manhattan distance

$$distance_{manhattan}(\vec{v}, \vec{w}) = \sum_{i=1}^{N} |v_i - w_i|$$

Now we can compute the Manhattan distance for *s1* and *s2*. Simply take the values in each dimension i from two vectors and sum their absolute values of difference. The distance is $|1-1| + |1-1| + |1-1| + |0-1| = 1$.

Euclidean distance

$$distance_{euclidean}(\vec{v}, \vec{w}) = \sqrt{\sum_{i=1}^{N} (v_i - w_i)^2}$$

For Euclidean distance, for each dimension i that two vectors have in common, we compute the squared differences between the values, sum them up and then take the square root of the sum. We get the Euclidean distance of *s1* and *s2* by $\sqrt{(1-1)^2 + (1-1)^2 + (1-1)^2 + (0-1)^2} = 1$.

For the difference between the Euclidean and the Manhattan distance, see Jurafsky and Martin [2009, 697-699] for a detailed review. They also pointed out that Euclidean and Manhattan distance metrics are not usually used for word similarity because they are sensitive to long vectors [ibid., 698]. However, these two metrics can be used to compute point distance in multidimensional scaling (MDS) in a computationally efficient way[13].

Dot product and cosine similarity Two widely used measures for word similarity developed in information retrieval and information theory are the dot product and the cosine similarity of two vectors [ibid., 698].

dot product or inner product

$$sim_{doc-product}(\vec{v}, \vec{w}) = \vec{v} \cdot \vec{w} = \sum_{i=1}^{N} v_i \cdot w_i$$

Back to our vectors for Example 2.2, the dot product of *s1* and *s2* is $1 \cdot 1 + 1 \cdot 1 + 1 \cdot 1 + 1 \cdot 0 = 3$. We normalize the dot product with the

[13]See a Python implementation with Euclidean distance as the dissimilarity measure at http://scikit-learn.org/stable/modules/generated/sklearn.manifold.MDS.html (accessed 20 May 2017).

vector length.

vector length

$$|\vec{v}| = \sqrt{\sum_{i=1}^{N} v_i^2}$$

Length (aka norm) of *s1* and *s2* can be computed by $\sqrt{1^2+1^2+1^2+0^2} = \sqrt{3}$ and $\sqrt{1^2+1^2+1^2+1^2} = \sqrt{4} = 2$, respectively.

normalized dot product (i.e. cosine)

$$sim_{cosine}(\vec{v}, \vec{w}) = \frac{\vec{v} \cdot \vec{w}}{|\vec{v}||\vec{w}|} = \frac{\sum_{i=1}^{N} v_i \times w_i}{\sqrt{\sum_{i=1}^{N} v_i^2}\sqrt{\sum_{i=1}^{N} w_i^2}}$$

Cosine similarity for *s1* and *s2* is $\frac{3}{\sqrt{3}\cdot\sqrt{4}} = 0.866$. The advantage of the cosine metric is that it circumvents the sensitivity in vector computation induced by the longer vectors after the normalization by vector length. Cosine similarity can take values ranging from $[-1, 1]$: 1 indicates the two vectors pointing to the same direction (high similarity); 0 means the two vectors are orthogonal (no common terms); -1 means the two vectors point to opposite directions (completely different) [ibid., 699]. We can see that 0.866 indicates that *s1* and *s2* are quite similar in meanings (which is not true, however!).

From this simple example, we learn that it is not a trivial question of how to represent word meanings computationally so that we can approximate the similarity indicated by commonsense knowledge of the world. We are in need of a representation that can better capture semantic and syntactic regularities in sequences.

Mitchell and Lapata [2008] proposed a framework for representing the meaning of phrases in a vector space by vector composition. Vector composition is of key importance to their approaches, which they operationalized in additive and multiplicative functions. They tested empirically the various composition methods on a sentence similarity task. Experimental results demonstrate that multiplicative models are superior to the additive alternatives when compared to human judgments.

Given two vectors v, w, let i denote the element-wise operation in v, w for the ith component, p the composed vector. α, β, γ are weighting constants. They confirmed the effective vector composition methods could be addition $p_i = v_i + w_i$, weighted addition $p_i = \alpha v_i + \beta w_i$, multiplication $p_i = v_i \cdot w_i$ and a combination of addition and multiplication $p_i = \alpha v_i + \beta w_i + \gamma v_i w_i$. It is worth noting that the strategies of vector composition they proposed can be applied to different types of textual

similarity measures.

2.3.3 Word Embedding Similarity

With the paradigm of distributional semantics left behind, embedding has recently become the buzz word in CL and NLP. According to Baroni et al. [2014], embeddings are referred as context-predicting methods because they are optimized through learning the contexts in which the words tend to appear, whereas context-counting methods initialize vectors with co-occurrence counts [ibid., 239]. Providing the extensive evaluation, the authors also point out that context-predicting models outperform a number of state-of-the-art context-counting models, e.g. Singular Value Decomposition (SVD see Golub and Van Loan [2012]), Non-negative Matrix Factorization (NMF see Lee and Seung [2001]), Latent Dirichlet Allocation (LDA see Blei et al. [2003]), in computing semantic similarity of words, phrases and sentences. The comparison by Baroni et al. [2014] has proven the effectiveness of word embeddings in modeling the semantic and syntactic context of words. The following subsections are devoted to the literature review on word embeddings and how word embeddings can expand the semantic features of words.

2.3.3.1 Word embeddings

The basic idea of word embeddings is to represent a word as a vector with real numbers in a vector space. There are many ways of creating vectors, amongst which the simplest method is the BoW representation with one-hot encoding (for more details see Rong [2014]). The one-hot encoding representation for *s2* in Example 2.2 is shown in Table 3. Each word in the sentence is represented in the dimension of itself as 1, with the other dimensions equal to 0. The dimensions for vectors are unique words in the vocabulary ("cat", "ate", "mouse", "food" in Example 2.2). The words in the columns are addressed in Table 2.2 as input words because the input format for training word embeddings as proposed in the models of `word2vec` [Mikolov et al., 2013a] is one-hot encoded.

Two important architectures of `word2vec` are continuous bag-of-words (*CBOW*) and *skip-gram* [Mikolov et al., 2013a,b]. For *CBOW*, one tries to predict the target word based on its context words. In *s2* of Example 2.2, if the target word is "ate", given a context window of three, the context words are one word on the left and right of the target word, namely, "mouse" and "cat". *CBOW* takes the sum of the vectors of the input context words. We move the context window along the whole sentence,

		dimensions			
		cat	ate	mouse	food
input words	*mouse*	0	0	1	0
	ate	0	1	0	0
	cat	1	0	0	0
	food	0	0	0	1

Table 3: One-hot encoding of *s2* in Example 2.2

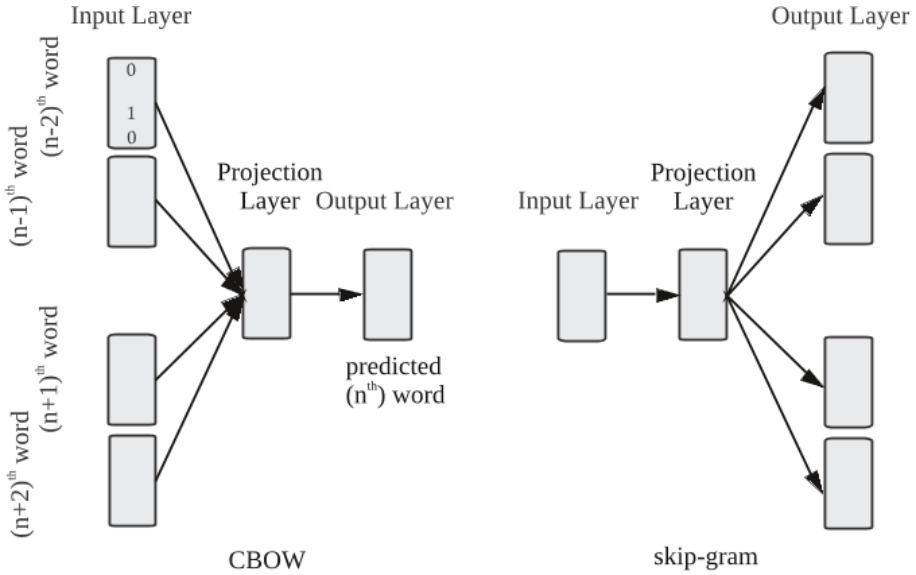


Figure 3: *CBOW* and *skip-gram* architectures in `word2vec` [Soutner and Müller, 2014, 152]

so in the next iteration, the target word will become "cat", with the context words "ate" and "food". *Skip-gram* is the opposite of *CBOW*, namely, it tries to predict the context of a word. In our example, given the word "ate", *skip-gram* predicts its context words "mouse" and "cat". Visualization of *CBOW* and *skip-gram* is shown in Figure 3 with a context window equal to five.

`word2vec` is a computationally efficient way of calculating word embeddings using *CBOW* and *skip-gram* models. It utilizes negative sampling which is a way of randomly sampling co-occurrences in a corpus (for more technical details, see Mikolov et al. [2013b, 3-4] and Rong [2014, 13]). Take the co-occurrence with the word "cat" in our imaginary corpus as an example. Instead of extracting all the words that follow "cat", we only sample a few words, e.g. "walk", "woman", etc. Negative sampling increases the computational efficiency to calculate word embeddings. As

mentioned previously, the length of word vectors is usually the length of the vocabulary, and these vectors are usually very sparse. To have a condensed representation of vectors, dimensionality reduction is necessary.

`word2vec` models implement shallow neural networks (a model for supervised learning, composed of an input layer, a hidden layer, an output layer and non-linear activation functions), which have only one hidden layer (for more, see Rong [2014]). A dimensionality reduction procedure takes place with the neurons in hidden layers. `word2vec` models are trained default with randomly initialized weights; the final embedding of a given word is the row vector of the weight matrix between the input layer and the hidden layer after several epochs of training[14]. Pretrained word embeddings with `word2vec` for English using *Google News* corpus (three billion words, see Mikolov et al. [2013a, 6]) are available online as `GoogleNews-vectors-negative300.bin.gz`[15]. The pretrained word embeddings contain three million 300-dimension English word vectors. We can use pretrained word embeddings in NLP tasks, or we can train models with pretrained word embeddings as weights for the words in the corpora of our own choice. Consequently, the resulting word embeddings can better represent the meaning of words from the domain of our chosen corpora.

Word embeddings enable us to perform vector-based calculations, e.g. to compute cosine similarity scores between words. The vectors of "cat" and "mouse" are represented as $[0.012, 0.204, -0.285, 0.217, 0.118, ...]$, $[0.240, 0.003, -0.101, 0.132, -0.034, ...]$ (values rounded to thousandths, only the first five dimensions out of 300 are shown), respectively.

Using pretrained *Google News* word embeddings, the cosine similarity between the vectors for "cat" and "mouse" is 0.466. In comparison with this, cosine similarity of "cat" and "dog" is 0.761, which corresponds to human intuition that words such as "cat" and "dog" tend to share similar contexts, while "cat" and "mouse" lie further away from each other in semantic and syntactic relations. Other interesting examples are the verb pairs "eat-drink" and "ate-drank": the latter pair has a higher similarity score of 0.599 compared to that of the former pair (0.507). The discrepancy in similarity scores can be explained by the multiple senses of "drink" as noun and verb, which might appear in various contexts, whereas "eat" can only act as a verb. On the other hand, "ate" and "drank" are both verbs in past tense; therefore, this pair shares under most circumstances similar contexts.

[14] See visualization in the section *Doesn't word2vec take in very different inputs from what is in wevi?* at https://docs.google.com/document/d/1qUH1LvNcp5msoh2FEwTQAUX8KfMq2faGpNv4s4WXhgg/pub (accessed 10 April 2017) on word embeddings (e.g. final vector products).

[15] https://code.google.com/archive/p/word2vec/ (accessed 24 April, 2017).

In summary, we can see that word embeddings are different from distributional representation of word meaning in terms of vector generation. However, they share the same measure of similarity, namely, cosine similarity. This is also the reason why the two terms are used interchangeably in a lot of literature without clearly distinguishing them. In this thesis, we explicitly clarify the distinction between distributional semantics and word embeddings. In the next section, we will discuss how to move from word embeddings to document embeddings, as we are interested in computing the distance between the entire sequences.

2.3.3.2 From word embeddings to document embeddings

Once we move from words to documents, word embeddings are useful in computing the document representation in the vector space constructed by words. There have been many discussions on document embeddings and attempts to derive document embeddings from word embedding. We provide hereby a comprehensive review of approaches to construct document embeddings.

Averaging word embeddings We can make use of word embeddings to arrive at document embeddings. There are two ways in general: average of word vectors, TF/IDF-weighted average of word vectors (for Python implementations, see Sarkar [2016, 188-193]). It has been proven effective that the vector representing a sentence is the centroid (i.e. the element-wise average) of the vectors of words that constitute the sentence [Sultan et al., 2015, 150]. This average vector will then represent the meaning of the whole sentence. In the second approach, we take the word vectors and multiply them with their TF/IDF scores. The TF/IDF score of a word can be computed from an available corpus (e.g. *Wikipedia*) or our training corpus.

These two approaches are easy to compute since they ignore word order, but for many applications, this is sufficient (especially for short documents). The efficacy of this operationalization is illustrated by Lilleberg et al. [2015]. The authors trained a multiclass classifier with element-wise TF/IDF-weighted word embeddings on the corpus *20 newsgroups*[16]. Word embeddings and TF/IDF scores were trained using the same corpus. For the classification of 20 topics, the TF/IDF-weighted word embeddings as features have reached an accuracy of 70%.

Document embeddings There are other techniques emerging from the deep learning community which computes the document embeddings directly. With

[16]http://scikit-learn.org/stable/datasets/twenty_newsgroups.html (accessed 10 May 2017).

the introduction of document embeddings (aka paragraph vectors), Le and Mikolov [2014] fully launched the discussion in the deep learning community about whether we can generate meaningful embedding representations for paragraphs.

The BoW model has the following two major weaknesses (1) the ordering of the words is lost; (2) the semantics of the words are ignored [ibid.]. A paragraph vector, an additional input in the neural network, can learn "fixed-length feature representations from variable-length pieces of texts" [ibid., 1188]. The authors proposed an algorithm to represent each document by a dense vector (e.g. embedding) and claimed that this vector could be trained to predict words in the document. The authors have pointed out that for a document which is composed of various paragraphs, each paragraph has its unique paragraph vector, while the word vectors are shared within the same document. A paragraph vector can be seen as a memorizing unit for contextual information encoded in a larger context (aka the whole paragraph), which is "the topic of the paragraph" [ibid., 1190].

The authors tested their algorithms on two tasks: sentiment analysis (by using paragraph vectors as features which are then fed into classifiers such as multi-layer perceptron (MLP, introduced in Section 2.4.4) and logistic regression[17]) and information retrieval (by calculating the distances between retrieved texts). Similar to `word2vec` which comes in two flavors, *CBOW* and *skip-gram*, paragraph vectors can also be used in two ways: the distributed bag-of-words model (*DBOW*) and the distributed memory of paragraph vector (*DMPV*) [ibid.].

Figure 4 shows *DBOW* which resembles the architecture in *skip-gram* because the word order in a paragraph is not considered. The input of *DBOW* is "a special token representing the document" [Lau and Baldwin, 2016, 79]. Figure 5 shows another approach in which a paragraph vector acts as an input unit together with other words sampled from that paragraph [Le and Mikolov, 2014]. Lau and Baldwin further specified the *DMPV* architecture which is similar to that of *CBOW* and concatenates vectors of a document token and multiple target words to predict a context word [ibid., 79]. Note that Le and Mikolov did not specifically test the efficacy of paragraph vectors in a classification task.

[17]Logistic regression is per se a softmax function for two classes. For more details, see Section 2.4.4.

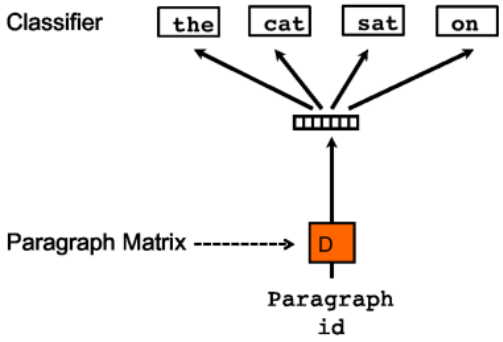


Figure 4: *DBOW* model of paragraph vectors [Le and Mikolov, 2014, 1191]

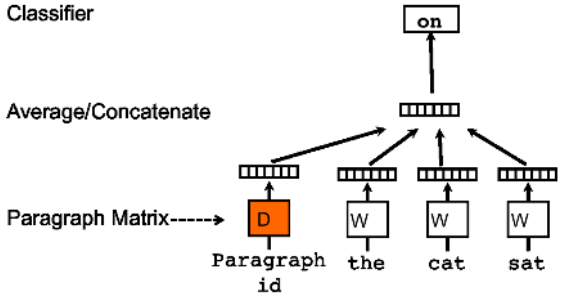


Figure 5: *DMPV* model of paragraph vectors [Le and Mikolov, 2014, 1190]

The Python library `gensim`[18] provides a wrapper for the models of paragraph vectors called `doc2vec`, with which a text snippet can be converted into a vector representation in the vector space constructed by words. This provides further possibilities to compare the paragraph vectors using similarity measures (e.g. cosine similarity).

Lau and Baldwin [2016] have further developed the `gensim` library with the possibility to use pretrained word embeddings in document embedding training[19]. This library is an extension to the existing `gensim doc2vec` libraries. In the updated `gensim` library by Lau and Baldwin [2016], we can use the Python classes `Doc2Vec` and `Word2Vec` to add pretrained word embeddings (e.g. those from *Google News*). This gives us retrained word embeddings customized to our corpus and domain.

The authors also tested the efficacy of distributed paragraph vector models

[18]https://radimrehurek.com/gensim/models/doc2vec.html (accessed 10 May 2017).

[19]https://github.com/jhlau/doc2vec (accessed 18 April 2017).

(*DBOW* and *DMPV*) in the tasks of pair duplication identification and semantic textual similarity. As *DBOW* ignores word order, the model is simpler compared with *DMPV* that has more parameters to train [ibid., 78]. After testing the `doc2vec` models in different task settings, they concluded that it is possible to improve `doc2vec` through careful hyperparameter optimization or with pretrained word embeddings [ibid., 85]. Moreover, they reported that for smaller corpora with short documents (13 tokens on average), the methods of averaging word embeddings worked better than the `doc2vec` models [ibid., 83], and *DBOW* worked better for longer documents (130 tokens on average) [ibid., 80]. Overall, they found that the `doc2vec` models could deliver better performance than the `word2vec` models in tasks where similarity computation is required, and that *DBOW* is a better model than *DMPV* in computing similarities of text snippets. Hence, they recommended that `doc2vec` models could be used as off-the-shelf models.

In this chapter, we have made clear the distinction between three types of textual similarities, i.e. surface lexical similarity, distributional semantic similarity and word embedding similarity. We have also discussed their measures and briefly touched upon how this can influence the quality of text categorization. In the next sections, we will systematically investigate the literature on text categorization and its correlation with text similarity.

Another interesting perspective on text categorization is the granularity of analysis, namely, unit of analysis in categorization, be it the whole document, the paragraph, or the sentence (for a comprehensive introduction to text classification on various levels of analysis using different techniques see Grimmer and Stewart [2013]). In the following three sections, an extensive literature review is provided on machine learning techniques which are applicable to various textual levels.

2.4 Supervised Techniques: Classifiers for Text Classification

In this section, we briefly summarize the basics of popular classifiers applicable to text classification: parametric (i.e. tunable parameters in classifiers, e.g. support vector machine (SVM), MLP, stochastic gradient descent (SGD)) and nonparametric models (data-driven learning, instant-based learning, e.g. k-nearest neighbor (KNN))[20].

[20]Raschka [2015, 93] provides a short description of parametric and nonparametric models.

2.4.1 K-nearest Neighbor (KNN) Classifier

KNN, a nonparametric model, does not "learn a discriminative function from the training data but memorizes the training set instead" (see Raschka [2015, 92-96] for more explanation and a Python implementation). Figure 6 illustrates how the KNN algorithm assigns instances given three classes (i.e. *minus, plus, triangle*)[21]:

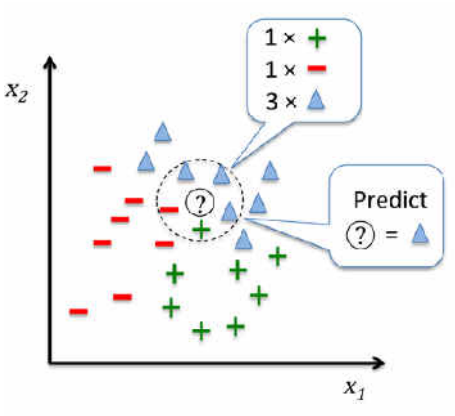


Figure 6: A simple example of a KNN classifier [Raschka, 2015, 93]

1. The algorithm locates a new instance (here the question mark in the dashed circle) in the feature space (created by features x_1 and x_2) based on a certain distance metric (e.g. Euclidean distance, see Section 2.3.2).

2. KNN chooses k nearest neighbors of the instance we want to classify based on a given k (here $k = 5$). In the example, we identify the following five instances as the nearest neighbors: three of the class *triangle*, one of the class *minus*, and one of the class *plus*.

3. KNN assigns the class label to the new instance by majority vote. The class *triangle* has the majority vote of three; thus, the new instance gets the label of *triangle* as its output.

KNN classifier does not involve a clear division of training and testing steps; therefore, it is also described as a "lazy learner". Its main advantage is the fast adaptation to new data instances, which comes with the downside that computational complexity grows linearly with the increasing data instances and vector dimensionality [ibid., 94].

[21] Raschka [2015, 93] lists three steps in KNN learning and we explain here concretely the three steps based on the example shown in Figure 6.

2.4.2 Support Vector Machines (SVMs)

Another powerful classifier is SVM, whose optimization objective is to maximize the margin, defined in Raschka [2015, 75] as "the distance between the separating hyperplane (decision boundary)". Support vectors are "the training samples that are closest to this hyperplane" [ibid., 75]. In Figure 7, the important concepts *hyperplane*, *margin* and *support vectors* are illustrated.

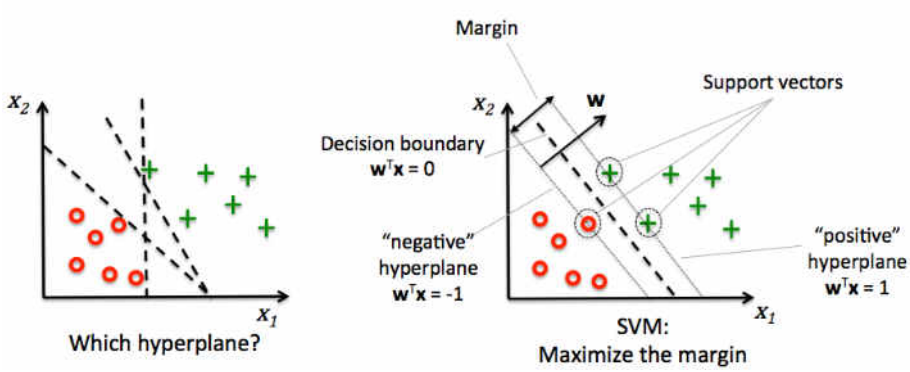


Figure 7: Illustration of a SVM classifier [Raschka, 2015, 75]

2.4.2.1 Linear and non-linear SVMs

Figure 7 demonstrates the ideal case of applying a linear SVM. It means that we would be able to separate samples from two classes (*circle* and *plus*) very well using a linear hyperplane as the decision boundary shown in Figure 7. In most of the real cases, however, we are confronted with non-linear separable classification problems. We, therefore, need another important variant of SVM, non-linear SVM, to tackle the non-linear decision boundaries as depicted in the top-left scenario in Figure 8.

The basic idea behind non-linear methods is to "create nonlinear combinations of the original features and to project them onto a higher dimensional space via a mapping function ϕ, where the data becomes linearly separable" [ibid., 75]. As shown in Figure 8, we can transform a two-dimensional dataset in a new three-dimensional feature space where the classes become separable via the following projection[22]: $\phi(x_1, x_2) = (z_1, z_2, z_3) = (x_1, x_2, x_1^2 + x_2^2)$. x_1 and x_2 are two features in a learning problem. z_1, z_2, z_3 are the transformed features from x_1, x_2, where $z_1 = x_1, z_2 = x_2$ and $z_3 = x_1^2 + x_2^2$. The new features z_1, z_2, z_3 construct a three-dimension space. This transformation can separate the two classes in the three dimensional space as

[22]The mathematical formula is taken from Raschka [2015, 76].

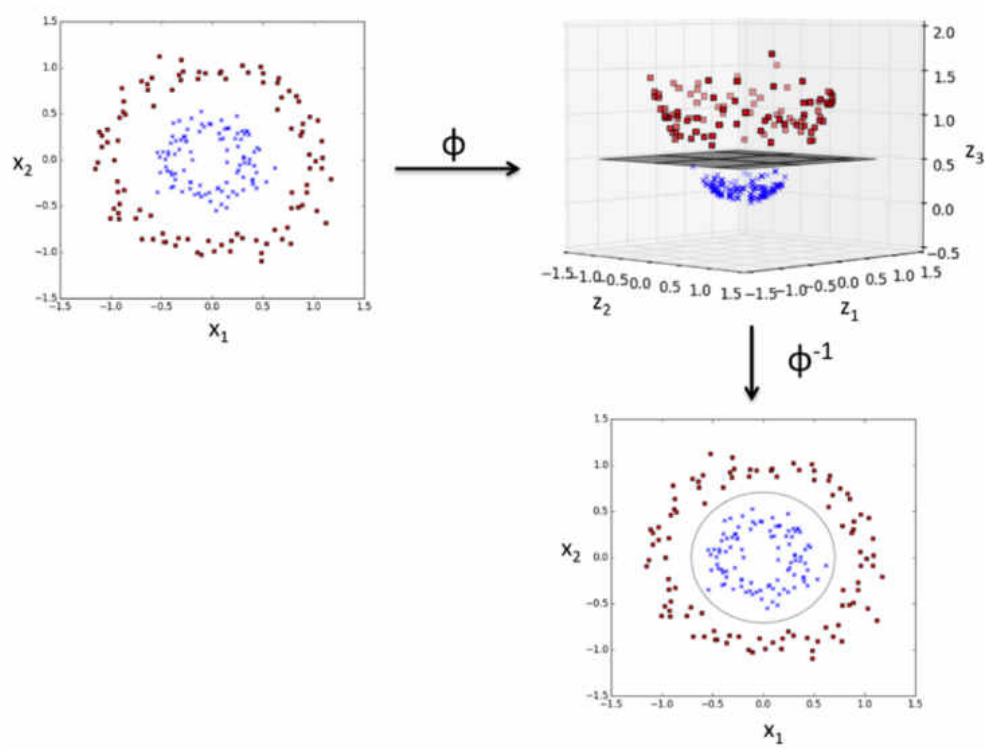


Figure 8: Illustration of a non-linear decision boundary and a non-linear SVM classifier [Raschka, 2015, 76]

shown in the top-right scenario in Figure 8, where the two classes are separable by a linear hyperplane. Once we project the two classes and their decision boundary from the three dimensional space back to the two dimensional space, we will obtain a circle as a non-linear decision boundary (see the bottom-right scenario in Figure 8). More explanation on linear and non-linear SVMs can be found in Raschka [2015, 75-80].

2.4.2.2 One-vs-one (OvO) and one-vs-the-rest (OvR) SVMs

A single SVM generates a decision boundary (linear or non-linear) between two classes. In order to extend binary classification to multiclass settings, we discuss two popular strategies (OvO and OvR) in applying SVMs in multiclass settings. The main difference between the two strategies is the number of classifiers to learn.

Discussions on the two strategies are covered extensively in the technical manuals of `scikit-learn` [Pedregosa et al., 2011], a machine learning library in Python

and the *StackExchange* online forum[23]. We briefly summarize the advantages and disadvantages of the two strategies based on the input from those discussions.

OvO Assume we have N different classes. OvO builds $\frac{N*(N-1)}{2}$ classifiers in total and trains one classifier for two classes each time. At prediction time, it then takes the majority votes of a class label as the predicted label. The computation is rather expensive (compared with OvR); however, it is less susceptible to an imbalanced distribution of classes because it uses the majority vote to determine the final label for an instance.

OvR This strategy is also known as *one-vs-all*, which fits one classifier per class by treating one class as "positive" and all other as "negative". It means: For class i, a classifier using OvR regards only i as the "positive" labels and the other labels as "negative". Hence, OvR strategy trains only N classifiers in total. Although it is computationally less costly, each classifier learns only a small subset of the data. The major disadvantage of the OvR strategy, therefore, is that it is sensitive to a skewed distribution of classes because the skewed distribution influences the numbers of instances in the "positive" and "negative" classes.

To summarize, for a multiclass classification task using SVMs, we have to consider factors such as the distribution of classes and the computational efficiency and opt for the suitable strategy. Thanks to the Python library `scikit-learn`, we can easily extend machine learning techniques and their evaluation metrics from binary settings to multiclass settings with OvO or OvR. More discussion on this matter can be found in Section 4.4 and Section 5.3.

An early work on multiclass legal document classification was conducted for Portuguese records of European Portuguese legal texts (around 8,000 documents). It has proven the efficacy of non-linear SVMs, as well as that of linguistic preprocessing, e.g. lemmatization, part-of-speech (PoS) tagging, etc. Linguistic processing can bring an increase of 5% to 10% for classification [Gonçalves and Quaresma, 2005]. Another classification of juridical documents (a multiclass classification, eight classes) was conducted by Pinto and Melgar [2016] as a comparison among four classifiers: KNN (see Section 2.4.1), SVM, naive Bayes (NB) and complement naive Bayes (CNB)[24].

[23]Discussions on OvO and OvR can be found at
https://stats.stackexchange.com/questions/91091/one-vs-all-and-one-vs-one-in-svm,
http://scikit-learn.org/stable/modules/multiclass.html,
http://scikit-learn.org/stable/modules/generated/sklearn.multiclass.OneVsOneClassifier.html,
http://scikit-learn.org/stable/modules/generated/sklearn.multiclass.OneVsRestClassifier.html
(accessed 10 May 2017).

[24]NB is a conditional probability model, which calculates the conditional probabil-

SVM was the winner with the Receiver Operator Characteristic area under the curve (ROC AUC)[25] equal to 84.6%, followed by KNN with 83.1%. The algorithms were trained with 5,471 documents from the attorney's office in Brazil in a 25-fold cross-validation setup. The comparison amongst classifiers confirms the efficacy of KNN and SVM for multiclass text classification.

Other experiments of text classification in the legal domain have been reported by de Maat and Winkels [2008, 2009, 2010], where the authors tried to classify sentences and provisions in Dutch legal documents with a rule-based system that learned from patterns and with a linear SVM. They firstly defined and annotated the sentence structures by the linguistic patterns in eleven classes, based on which the rule-based system made predictions. The linear SVM, on the other hand, learned from the features computed by the BoW model with TF/IDF weights or with binary values. To compare the efficacy of two systems, around 600 legal sentences were tested. The authors showed that both of the systems had reached an accuracy of 90%. Furthermore, the rule-based learner was better than the SVM (3% higher in accuracy). The authors concluded that the rule-based system is more concise and accurate, yet SVM has a higher adaptability for unseen instances in training set. In addition, one can quickly adapt machine learning systems to tackle a large number of new texts, while rule-based systems take longer to develop because one has to define new rules tailored to the new texts.

2.4.3 Stochastic Gradient Descent (SGD)

Raschka [2015, 34-47] and the `scikit-learn` manuals[26] offer detailed explanations on the SGD classifier. We summarize their input as follows. SGD is a linear classifier (see Section 2.4.2.1 for the discussion on linearity and non-linearity). The SGD classifier uses gradient descent to optimize the weights that minimize cost function in classification [Raschka, 2015, 34]. *Stochastic* here means online learning where

ity $p(C|x_1, ..., x_n)$, given a class C and some n features represented by $\boldsymbol{x}$ (see https://en.wikipedia.org/wiki/Naive_Bayes_classifier (accessed 20 March 2017)). For a more detailed review on NB classifier see Raschka [2014]. The complement naive Bayes (CNB) classifier has been firstly introduced by Rennie et al. [2003] which tackles the problem of skewed class distribution in a multiclass classification task. CNB is trained using data in all other classes except the one which we are interested in [ibid., 618].

[25] ROC AUC refers to the area under the ROC curve. Raschka [2015, 193-197] gives a comprehensive introduction to this metric, which evaluates classifiers by calculating the ratio of the false positives and the true positives. It has been shown that ROC AUC and accuracy correlate well in evaluating classifier performance [ibid., 197].

[26] See http://scikit-learn.org/stable/modules/sgd.html, http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.SGDClassifier.html (accessed 10 May 2017).

the algorithm is optimized incrementally as new training instances arrive [ibid., 43].

An important hyperparameter for the SGD classifier is *learning rate*, a constant within the range $[0, 1]$, which defines the step we make in the optimization in each iteration. As we do not want to *overshoot* the global minimum in cost function [ibid., 40], we usually apply a decreasing learning rate. Other tunable hyperparameters in the SGD classifier include the cost function, the iteration times, the penalty, the momentum, etc[27]. Parameter tuning in the SGD classifier is a double-edged sword: It gives the classifier high flexibility in learning at the cost of its sensitivity to the various possible combinations of hyperparameters.

2.4.4 Multi-layer Perceptron (MLP): Shallow Neural Network

Multi-layer shallow feedforward neural network is also called MLP, which refines the architecture of linear SGD classifiers by replacing its linear activation functions with non-linear ones. MLP is composed of one input layer, one hidden layer, and one output layer. Since it has only one hidden layer, it is not considered a deep learning network [Raschka, 2015, 345]. We see from a simple graphic illustration of MLP shown in Figure 9, the network is a fully connected neural network. MLP conducts its supervised learning task and finds the optimal sets of network weights through the following operations (adapted from Raschka [2015, 344-350]):

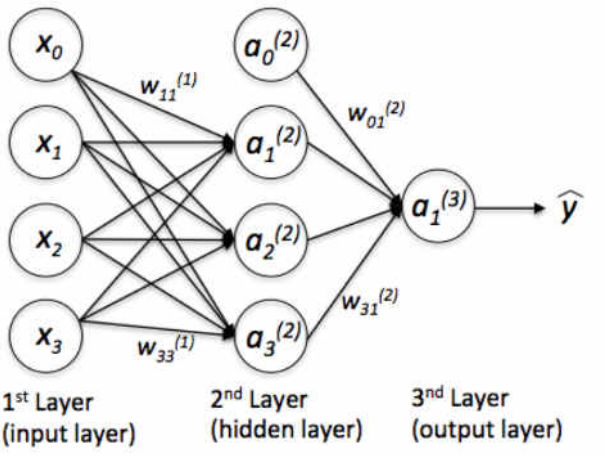


Figure 9: Illustration of a MLP classifier [Raschka, 2015, 345]

1. We start at the input layer (x_1, x_2, x_3 in Figure 9) with vectors[28]. In textual analysis, we usually use one-hot encoded vectors (see Section 2.3.3) to represent

[27] Refer to http://scikit-learn.org/stable/modules/sgd.html (accessed 10 May 2017) for the explanations on parameters and code examples.

[28] x_0 is a biased term which avoids the network receiving zero input.

the input words. The patterns of training instances are fed forward through the network to generate output $\hat{y}$ with randomly initialized weights $w^{(1)}, w^{(2)}$.

2. Based on the network's output $\hat{y}$, we calculate the error $(y - \hat{y})$ that we want to minimize using a cost function (e.g. logistic cost function).

3. We then update the model by backpropagating the error and finding its derivative for each weight in the network. It means: We "compute the gradient based on the whole training set and update the weights of the model by taking a step into the opposite direction of the gradient" [ibid., 344] of cost function respective to the weights.

4. We repeat the previous steps for multiple epochs and let the MLP learn the weights. At prediction time, we use the learned weights and forward propagation to calculate the network output and apply a softmax function to output "the predicted class labels in the one-hot representation". It means: In the output vector, only the dimension of the predicted class has the value 1; the other dimensions have the value 0.

MLP makes use of non-linear activation functions to outperform the traditional linear classifiers such as linear SVM and SGD. Popular non-linear activation functions are logistic (*sigmoid*), hyperbolic tangent (*tanh*) and rectified linear unit ($ReLU$).

sigmoid

$$f(x) = \frac{1}{1 + e^{-x}}$$

tanh

$$f(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

ReLU

$$f(x) = max(0, x)$$

Popular cost function used in MLP is *logistic loss* (aka *log loss* or *cross-entropy loss*):

$$-logP(y|\hat{y}) = -(y * log(\hat{y}) + (1 - y)log(1 - \hat{y}))$$

We often use the natural logarithm with base e.

The *softmax* function is "a generalization of the logistic function that allows us to

compute meaningful class probabilities in multiclass settings" [Raschka, 2015, 404]. It provides us with a normalized probability distribution of classes.

2.4.5 Convolutional Neural Network (CNN)

CNN is a kind of deep learning network applicable to text classification, and it consists of two particular strategies in its implementation: convolution and pooling. *Convolution* means the extraction of features from the grid-like input with the defined feature filters. *Pooling* is also called subsampling or downsampling, where we "reduce the dimensionality of each feature map while retaining the most important information"[29]. When applying max pooling, we simply take the largest value within the feature maps we define.

Kim [2014]; Zhang and Wallace [2015] explored extensively CNN (a deep learning neural network) as a classifier for text classification task (e.g. question classification, opinion classification, sentiment classification). The CNN architecture is composed of an embedding layer, followed by a convolutional, a max-pooling and a softmax layer. The embedding layer serves as the input to CNN and is calculated from the training corpus (i.e. task-specific embeddings). Essentially, the embeddings are co-occurrence matrices where each row is one token, and each column is the context word that co-occurs with the token. Embeddings as input can be learned from the model, with some real numbers initialized randomly. The network will then try to predict a label for the current input based on its compositional words. The errors are backpropagated to tweak the weights of the input words (embeddings). In the convolution layer, various feature filters are applied to the grid-like embedding matrices, and strides can be defined vertically and horizontally for the sliding directions and steps. The activation function of the convolution layer is *ReLu*. As a result, the embedding matrices are transformed into more condensed representations of texts. The 1-max-pooling strategy has been found to be the most successful pooling method. The *softmax* function is applied at the penultimate layer of the network to generate predictions on classes. The loss function is cross-entropy.

For a rich visualization of this architecture see Figure 10 which depicts the layers and their connections explicitly. Starting with the input sentence "I like this movie very much!", we represent the words in this sentence with a matrix in which the columns represent word embeddings (dimensionality equal to five). Then we extract the features with six feature filters of three different region sizes (i.e. $2\times5, 3\times5, 4\times5$),

[29]The paragraph is summarized from the input at https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/ (accessed 20 May 2017).

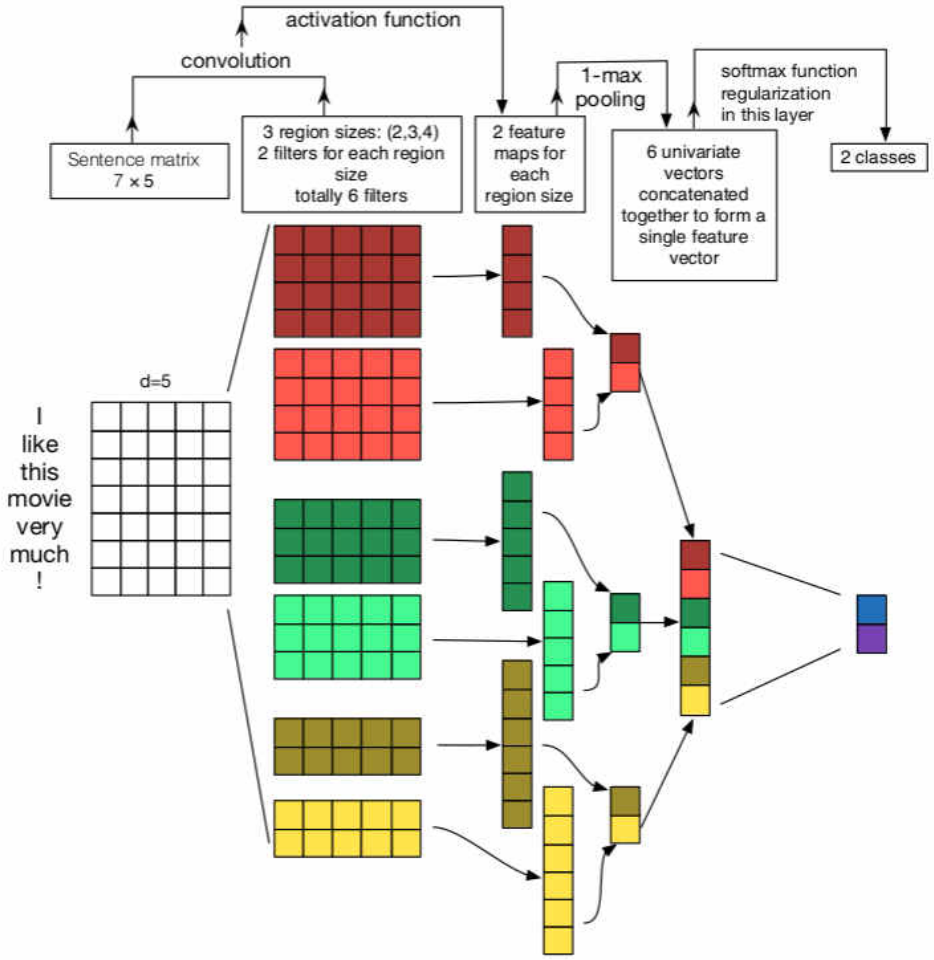


Figure 10: Illustration of a simple CNN for text classification [Zhang and Wallace, 2015, Figure 1]

with vertical strides equal to one. During the convolution, we take the dot product of each feature filter and the scanned text region and feed them into the activation function *ReLu*. Therefore, for the first filter in Figure 10 (marked in dark red, region size 4×5), we obtain four values after the convolution. In total, we arrive at six more condensed vector representations of the input matrix in the convolution step. In the 1-max-pooling step, we take the largest value in each region map. For the prediction of the class label, we apply softmax to the concatenated vector of the maximum values in the region maps.

Zhang and Wallace [2015] discuss hyperparameter tuning and its influence on the classification accuracy extensively. The most important parameters are input word vectors, the number of filters, filter region size, feature maps, strides, activation

function, pooling strategy, dropout rate and L2 norm constraint. Code examples have been provided on *GitHub* project `cnn-text-classification-tf`[29] for replication and further training. As the output labels of CNN are one-hot encoded, it is relatively easy to extend the architecture from binary classification to multiclass.

2.5 Unsupervised Techniques: Clustering and Topic Modeling

Unsupervised techniques vary largely from one to another: As long as we do not have pre-labeled materials in our training samples, we are dealing with unsupervised settings. As one of the earliest works for unsupervised learning in the legal domain, Merkl and Schweighofer [1997]; Schweighofer et al. [2001] applied a self-organizing map, an unsupervised neural model, to cluster international treaties from various domains.

Document clustering can be applied as a semantic compression strategy, i.e. to compress semantically and syntactically close documents together. K-means clustering is the most widely used clustering technique that works well for general purposes and not too many clusters[30]. Since clustering is one kind of unsupervised learning techniques, where the ground truth of test data is not available, proper evaluation of clustering is needed. In this section, we introduce two popular unsupervised approaches, i.e. k-means clustering and topic modeling.

2.5.1 K-means Clustering

This part provides a review of k-means clustering algorithm and the evaluation measures of clustering results. K-means clustering uses a set of k centroids; the *centroid* of a cluster is the average of that cluster, around which similar data points group. The simplest k-means starts with a random set of seeds selected from the training data and iteratively assigns the other data points by similarity [Raschka, 2015, 312-313]. The similarity within each cluster is measured by inertia, i.e. within-

[29] https://github.com/dennybritz/cnn-text-classification-tf (accessed 05 Jan 2017).

[30] For a detailed comparison among the clustering techniques, see http://scikit-learn.org/stable/modules/clustering.html (accessed 01 May 2017).

cluster sum-of-squares[31] defined as

$$\sum_{i=0}^{n} \min_{\mu_i \in C}(||x_j - \mu_i||^2)$$

μ_j denotes the mean (centroid) in the cluster, C a given cluster, x_i each instance that belongs to the cluster C, n the total number of instances in the cluster C. Inertia measures the least sum of squares of the distance between the cluster members and the cluster centroid, and this measure can be interpreted as "how internally coherent clusters are"[32].

As an input hyperparameter in K-means, the number of clusters is required to be specified. Therefore, it is worth noting that the disadvantage of k-means clustering is its sensitivity "to the initial set of seeds picked during the clustering" [Aggarwal and Zhai, 2012, 94]. Also, to avoid the "curse of dimensionality where the feature space becomes increasingly sparse for an increasing number of dimensions of a fixed-size training dataset" [Raschka, 2015, 96], it is recommended in practice to conduct dimensionality reduction or to feed the k-means algorithm with condensed vector representations[33]. For more on the training steps and a Python implementation of k-means clustering see ibid., 312-317.

To evaluate k-means clustering, either we make use of human-generated gold standards, or we calculate the Silhouette coefficient which does not require the gold standards. This metric is calculated by "using the mean intra-cluster distance and the mean nearest-cluster distance for each sample"[34]. For a set of samples, it is then computed as the mean of the Silhouette coefficient of each sample.

2.5.2 Topic Modeling

Topic modeling can be categorized under unsupervised learning because it makes use of statistical methods, without annotation data, to analyze "the words of the original texts to discover the themes that run through them, how those themes are connected, and how they change over time" [Blei, 2012, 77-78]. We make the following assumption for topic modeling: (1) *topics* are the hidden themes in documents;

[31] http://scikit-learn.org/stable/modules/clustering.html#k-means (accessed 01 May 2017).

[32] http://scikit-learn.org/stable/modules/clustering.html#k-means (accessed 01 May 2017).

[33] See in the Section 2.3.2 *K-means* from the `scikit-learn` manual at http://scikit-learn.org/stable/modules/clustering.html (accessed 01 May 2017).

[34] http://scikit-learn.org/stable/modules/generated/sklearn.metrics.silhouette_score.html (accessed 10 May 2017).

(2) a topic is approximated by a cluster of words which often co-occur; (3) a document may cover multiple topics; (4) a corpus exhibits the same set of topics [ibid., 83]. Topic models use generative probability models (e.g. LDA) which describes the probability of generating the given words on certain topics. Topic models maximize the probability of $p(topic|document) \times p(word|topic)$. The probability distribution can be derived iteratively by processing the corpus and measuring co-occurrence.

To generate representative keywords for each topic, topic models take very large corpora to function. See the review from Blei [2012] on probabilistic topic models for more details. In practice, it is quite common to use topic models for keyword generation after document clustering[35]. Topic modeling is therefore regarded as "a more general probabilistic framework which determines word clusters and document clusters simultaneously" [Aggarwal and Zhai, 2012, 100].

2.5.3 Summary of Unsupervised Techniques

Based on a brief review of two major unsupervised learning techniques, we can draw the conclusion that techniques in this category are quite diverse, which has its advantages and disadvantages. Unsupervised methods show advantages over supervised methods in its flexibility of method combination. So far it has been very promising to combine supervised classifiers and unsupervised clustering, as well as to combine two unsupervised methods as presented in Aggarwal and Zhai [2012]. The main limitation of unsupervised learning lies in the difficulties of evaluation, as we usually do not have human-generated gold standards to gauge the performance of systems. Another pitfall of unsupervised methods is from its data-driven nature, as we expect those techniques to unveil hidden structures of data that are hard to detect by human observations. It requires the researchers to acquire a general understanding of the data in use before one starts running unsupervised models on it blindly.

2.6 Semi-supervised Learning and Text Categorization

A setting of semi-supervised learning in text categorization refers to any setup that utilizes a small amount of labeled training material and a large amount of unlabeled data, to assign labels to the unlabeled material [Sammut and Webb, 2011, 897]. Aggarwal and Zhai [2012, 94] explicitly pinpoint the applicability of partial supervision,

[35] For an interesting code example, see the section *Latent Dirichlet Allocation* in *Document Clustering with Python* at http://brandonrose.org/clustering (accessed 10 May 2017).

one type of semi-supervised techniques, in document clustering using the k-means algorithm. Recall from the discussion in Section 2.5.1 that k-means is quite sensitive to the initialization of seeds; therefore, partial supervision helps the formation of coherent clusters in that we can assign a particular initial set of seeds as the centroids around which the final clusters can be formed [ibid., 94]. This semi-supervised technique has proven to be successful in influencing clustering of documents relating to a coherent subject matter with a pre-defined organization scheme of information. Aggarwal et al. [2004] showed the advantage of using partially supervised clustering to categorize heterogeneous collections of web documents. The authors proved that a priori knowledge of the definition of each category (aka a representative set of keywords) could increase the accuracy of categorization effectively. Under the settings of k-means clustering, the authors used as initial centroids the concatenation of the TF/IDF-weighted terms in documents of the given categories (e.g. wine, fitness, etc.) from a pre-existing taxonomy. This experiment shows a powerful semi-supervised approach to acquire a priori knowledge from the pre-existing knowledge bases and apply the knowledge to document clustering.

2.7 Summary

We have reviewed three different types of machine learning techniques and measures of text similarities that influence the data structure and algorithm we choose. It is also obvious that surface similarity of words and documents are not sufficient to represent meaning embedded in texts. Therefore, the experiments with the Jaccard coefficient proposed by Alschner and Skougarevskiy [2015, 2016a,b] on the level of IIA treaty articles cannot represent the semantic similarity of texts sufficiently, even if they managed to identify the identical segments in the treaty articles. To represent the meanings of treaty articles, we need to utilize methods from distributional semantics and word embeddings to generate vector representations as features in text categorization. The design of text categorization methods should be geared towards our corpus in this thesis. For this reason, we introduce our corpus and the preprocessing steps to extract titled and untitled articles in Chapter 3.

3 Data: SNIS English Corpus, IIA Treaties and Treaty Articles

In this chapter, we present our corpus of IIAs created by the SNIS project *Diffusion of International Law* (SNIS corpus hereafter). We then discuss the preprocessing steps, the problems and our solutions in extracting treaty articles from the SNIS corpus.

3.1 Corpus

The SNIS English corpus was created by converting "a broad variety of formats" [Sugisaki et al., 2016, 203] (e.g. PDF, HTML, *Microsoft Word*, etc.) and into XML documents and automatically translating non-English treaties into English. The original source files were provided by the legal experts and economists in the SNIS team, more details on source formats can be found in Alschner and Skougarevskiy [2016a]. In total, the corpus has 2,823 English treaties.

HTML documents account for a large portion of source formats (71.13%), which were collected from *Kluwer Arbitration*[1]. About 26% of the treaties were originally in PDF formats. The rest of the source files existed in *Microsoft Word* which are often entangled with code-switching lines (i.e. bilingual documents) as shown in Sugisaki et al. [2016, Figure 1, 205].

Dr. Kyoko Sugisaki has firstly converted the PDF documents into XML markup documents of treaty layouts (e.g. text blocks and paragraphs, see Sugisaki et al. [2016, 205] for more details) with *Abbyy Recognition Server*[2]. She then transformed the XML documents of layouts (*layout* XML hereafter) into structured XML documents (*content* XML hereafter) where basic treaty structures such as preface, preamble, body, chapter (including titles and texts), article (including titles and texts) have

[1] http://www.kluwerarbitration.com/ (accessed 20 May 2017).

[2] https://www.abbyy.com/en-eu/recognition-server/ (accessed 20 May 2017).

been properly assigned to various XML tags. Despite around 5% of Optical Character Recognition (OCR) errors[3], the XML quality is good enough for further automatic content analysis.

The original treaties were in 29 different languages, with English accounting for 72.7% (2,065 treaties), followed by French (290), Spanish (176), Arabic (89) and Russian (84). The rest of files in other source languages (119) takes up 4.21% in the corpus. Figure 11 presents the highly skewed distribution of source languages. Two-letter codes of languages in ISO639-1[4] are listed alphabetically in x-axis.

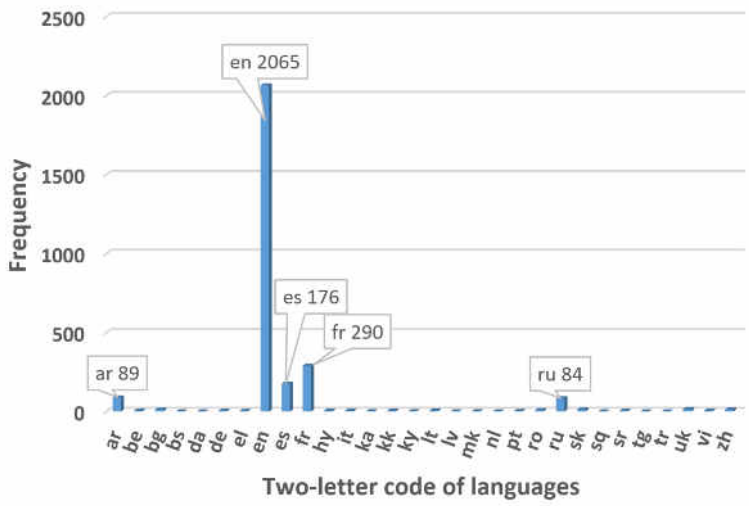


Figure 11: Distribution of treaties across source languages

In order to gather a complete picture of IIAs, non-English treaties were translated into English using statistical machine translation (SMT) systems (see Koehn [2009]). French and Spanish treaties were translated with in-house systems developed by the Institute of Computational Linguistics (the SNIS translation team hereafter). Lacking data in IIAs, the SMT systems were trained with large corpora of Europarl[5] [Koehn, 2005] and *JRC-Acquis* corpus[6] which include a large portion of legal texts. Additionally, the language models of the in-house SMT systems were trained with

[3]Dr. Kyoko Sugisaki has randomly selected three PDF documents to evaluate OCR, with clear text, blurred text, and bilingual text, respectively. The quality of conversion into XML drops when we move from clear text to bilingual text. The most common OCR error types are punctuations symbols, special characters, line breaks, handwriting and word recognition.

[4]https://en.wikipedia.org/wiki/List_of_ISO_639-1_codes (accessed 10 May 2017).

[5]The *Europarl* parallel corpus is extracted from the proceedings of the European Parliament. For download, see http://www.statmt.org/europarl/ (accessed 20 May 2017).

[6]"The Acquis Communautaire (AC) is the total body of European Union (EU) law applicable in the the EU Member States. This collection of legislative text changes continuously and currently comprises selected texts written between the 1950s and now." For more see https://ec.europa.eu/jrc/en/language-technologies/jrc-acquis (accessed 20 May 2017).

the in-domain English texts on IIAs. The SNIS translation team reported that the performance of our in-house systems only marginally outperformed *Google Translate* in translating French and Spanish IIAs. For treaties in other source languages, they were translated into English by *Google Translate*. It is reported that since there are five source languages for which *Google Translate* does not even provide the translation service, the SNIS translation team did not include them in our final version of SNIS English corpus[7].

Amongst 2,823 treaties, we can identify 58 distinctive contracting years with a peak from 1992 to 2004, as Figure 12 illustrates. In 1996 alone, 192 IIAs were negotiated. The growth of IIAs corresponds to the trend described in the UNCTAD report *Recent Developments in International Investment Agreements*[8].

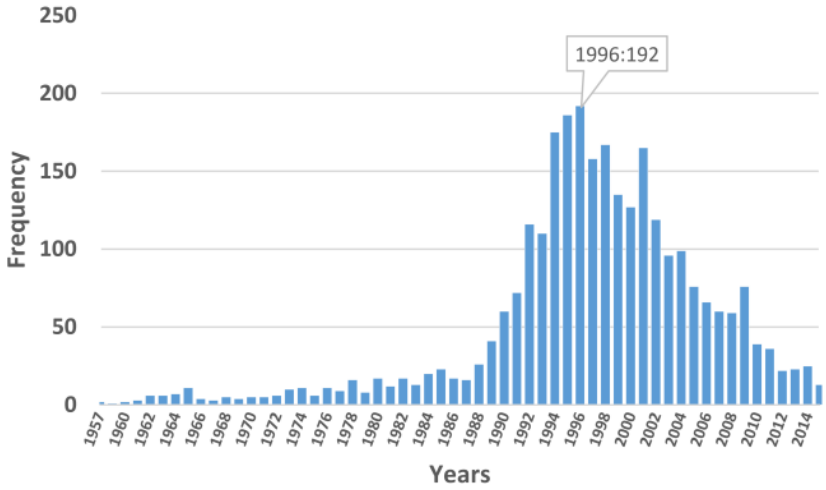


Figure 12: Distribution of treaties across years

We can identify 212 distinctive contracting parties as shown in Figure 13. We sort the three-letter codes of the contracting parties alphabetically as defined in ISO 3166-1[9] in x-axis. Due to the limited space, only the top ten active contracting parties in terms of the contracted IIAs are denoted in Figure 13: China (143 treaties), Switzerland (123), USA (107), UK (104), the Netherlands (102), South Korea (103), Turkey (100), Egypt (105), France (99) and the Belgo-Luxembourg Economic Union[10] (95). The three-letter codes, their corresponding contracting parties and the counts of negotiated treaties are listed in Table 21, Table 22 and Ta-

[7]At the time of writing, no concrete information on these five languages has been made clear. For more explanation on MT, refer to the SNIS translation team for assistance.

[8]http://unctad.org/en/docs/webiteiit20051_en.pdf (accessed 20 May 2017).

[9]For the mappings between the codes and parties at https://en.wikipedia.org/wiki/ISO_3166-1_alpha-3 (accessed 10 Jan 2017).

[10]BLEU in Figure 13.

ble 23 in Appendix A. Although almost all the economies around the globe (be they countries, organizations or regions) have participated in IIAs, based on the numbers of contracted IIAs, we see that the commitment of contracting parties varies largely. The IIAs which the top ten parties have negotiated takes up 38.3% of the treaties in our SNIS English corpus.

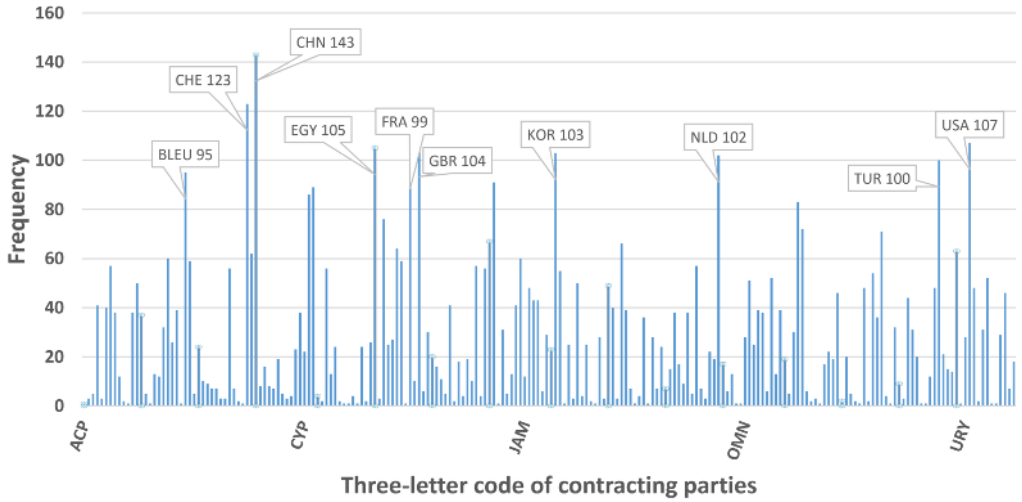


Figure 13: Distribution of treaties across contracting parties

category	name	source language	source format	treaty structure in XML
1	*EN_HTML_STRUCTURED*	English	HTML	structured
2	*EN_PDF_SEMI*	English	PDF	
3	*GOODMT_MIXED_SEMI*	other (good translation)	HTML, PDF	semi-structured
4	*BADMT_MIXED_SEMI*	other (bad translation)	PDF, HTML	

Table 4: Categorization of treaties in the SNIS corpus based on three criteria

In order to facilitate the extraction of treaty articles, we divide the treaties into four categories (*EN_HTML_STRUCTURED*, *EN_PDF_SEMI*, *GOODMT_MIXED_SEMI* and *BADMT_MIXED_SEMI*)[11], based on (1) the quality of translation, (2) the source format and (3) the treaty structure in XML. As shown in Table 4, the four categories can be distinguished based on three criteria:

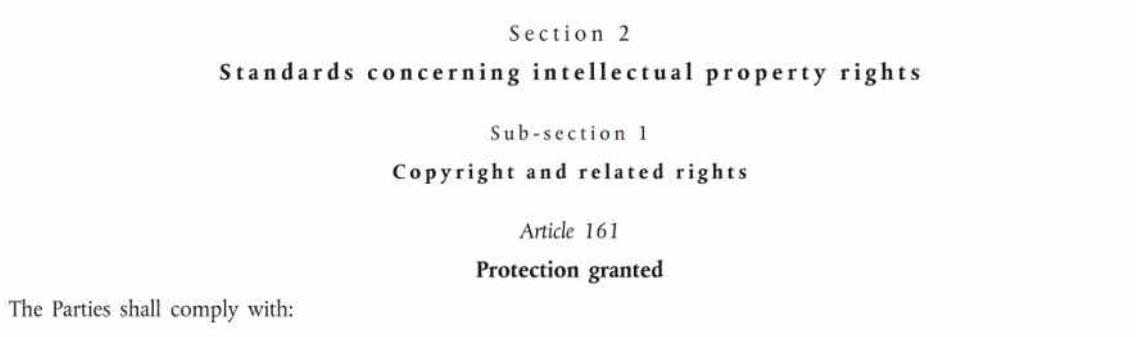
1. The quality of translation
   As the treaties were originally in various languages, in the SNIS corpus we either have treaties written in English or treaties translated into English. If the source treaty was already in English, the text should be of a high quality for the tasks later on. If the treaty has been translated from foreign languages, the

[11]Samples treaties in four categories in Appendix B.1.

translation quality of French and Spanish is generally satisfactory[12]. Translated treaties that were originally in Arabic have generally the worst quality, either due to the OCR quality of PDF documents or due to the poor performance of *Google Translate* on the language pair Arabic → English.

2. The source format
   Source file formats PDF and HTML account for 97% of the treaties in our corpus; therefore, we mainly differentiate between treaties that have been converted into XML from PDF and those that were in HTML. Treaties that originally were in PDF documents are prone to OCR errors due to various reasons, such as bad image quality, handwriting, bilingual texts, interletter spacing, uneven line spacing, special formats.

   Our corpus suffers from the letter spacing problems in titles because for article titles and article texts, the interletter spacing varies. It is very common that the article titles are formatted in monospaced bold fonts and with "loose" interletter spaces, while the article texts are with proportional fonts and normal interletter space[13]. For a snapshot of PDF shown in Figure 14, "loose" interletter spaces were turned into a fragmented text snippet "S t a n d a r d s c o n c e r n i n g i n t e l l e c t u a l p r o p e r t y r i g h t s" because the section title was spaced more loosely compared to its texts. We calculated the error


Section 2

Standards concerning intellectual property rights

Sub-section 1

Copyright and related rights

Article 161

Protection granted

The Parties shall comply with:

Figure 14: Interletter spacing in PDF

rate due to the interletter spacing in our corpus. The problematic segments were extracted by the following rule: Within a given block of texts, find all the longest consecutive sequences of single letters. This is an effective strategy

[12]Besides the feedback from the SNIS translation team, we have manually randomly inspected three translated treaties from French and Spanish, respectively, and can, therefore, conclude that the translation quality for French → English, Spanish → English has been satisfactory.

[13]For a detailed introduction to monospaced and proportional fonts in OCR, see *Fixed and Proportional Fonts* at http://www.how-ocr-works.com/OCR/word-character-segmentation.html (accessed 19 April 2017).

because we only have three English one letter words ("A", "I", "O"[14]) Out of all the titles and texts, we can identify 424 segments of consecutive single letters which take up 0.95% of all the text blocks.

From the perspective of OCR accuracy across languages, the quality of Arabic documents is expected to be the least pleasing, due to the facts that (1) characters are joined in Arabic with only a few letters disjoined; (2) the shape of some printed letters can be elongated in order to justify word segmentation[15]. It is also due to the poor OCR quality that the translated English treaties are barely readable which were originally in Arabic. The treaty "{GIN,TUN}_1990-11-18"[16] signed between Guinea and Tunisia was originally in Arabic. Its English translation in XML form is shown in Listing 3.1, which is hardly legible as proper English texts.

```
<?xml version="1.0"?>
<!-- SnisDocXML -->
<!--source:ABBYY-->
<!-- Translated by google translate into:en -->
<treaty>
<main language="ar">
<preface>
<p>Tuesday - 20 is the argument 1411-2 Jobb sheltered</p>
</preface>
<preamble><p>Newly released for Almtabaadalrsaid Republic
    Altoshid</p></preamble>
<body>
<p>The separation of a single - opening His Facebook Almmadah
    Al Almlhvh Alatphalah this Altanon and Almmermh Prague on
    March 14, 1990 between the Government of the Republic of
    Tunisia and the Government of the Federal Republic and the
    Czech Asalonakah and raw Ptgadi taxation Ghuraibi Ouchaa
    gypsum evasion with respect to income bedding Al ibex wealth
    . ...</p>
<p>Tawanan state.</p>
<! -- ... -->
</body></main></treaty>
```

Listing 3.1: English translation of an original Arabic treaty in XML

3. The treaty structure preserved in XML
   The *content* XML documents have been converted from the *layout* XML documents based on the mapping between the layouts (e.g. text blocks) and the treaty elements (e.g. article title, article text). As a result, we expect the varying mapping quality in the *content* XML documents, in terms of their source

[14] https://en.wiktionary.org/wiki/Category_talk:English_one_letter_words (accessed 20 March 2017).

[15] http://www.how-ocr-works.com/OCR/word-character-segmentation.html (accessed 19 April 2017).

[16] In the SNIS English corpus, we denote each treaty name with the following format: {party$_1$, ..., party$_n$}_year-month-day.

formats (i.e. PDF, HTML). In a *content* XML, we can expect the treaty element structure to be either fully preserved (*structured*), or partially preserved (*semi-structured*). Category 1 (*EN_HTML_STRUCTURED*) has a high correspondence between treaty structure and XML structure, i.e. the hierarchical structure of an IIA (e.g. preamble, body, chapter, article, title, text, conclusion, annex) has been properly transformed into XML tags and attributes. It is because those XML files have been converted from HTML documents. Category 1 does not share the XML structure with the other three categories. Listing 3.2 shows two examples of untitled articles in category 1. The titles (if any) should be stored under the XML tag `<title>` (line 2, 9). According to the condition whether the XML tag `<title>` is empty or not, articles can be divided into *titled* and *untitled* ones.

```
1  <article xml:id="1">
2  <title/>
3  <number>Article 1</number>
4  <p>For the purposes of this Treaty</p>
5  <p>1. the term "investments" comprises every kind of asset, in
       particular:</p>
6  <! -- ... -->
7  </article>
8  <article xml:id="2">
9  <title/>
10 <number>Article 2</number>
11 <p>(1) Each Contracting State shall in its territory promote as
        far as possible investments by investors of the other
       Contracting State and admit such investments in accordance
       with its legislation.</p>
12 <! -- ... -->
13 </article>
```

Listing 3.2: XML structure of category 1

Categories 2, 3 and 4 can have various possible structures in XML documents as shown in the six scenarios in Listing 3.3. Based on different XML structures with which the titles can be stored, treaties can be divided into two types, i.e. *titled* and *untitled*. We define the *titled* treaties with the criterion: whether an title and its corresponding text can be retrieved by our extraction algorithm. Scenarios 1 and 2 are with chapter or article titles as the *title* attribute of `<div>` tags. Scenarios 3, 4, 5 and 6 are either "authentically" or "formally" *untitled* titles. "Authentically" untitled articles (Scenario 6 in Listing 3.3, Figure 2) are those text blocks where no titles were given in the original source files, while "formally" untitled articles are those text blocks in which the titles did not end up in the `<div>` attribute *title* during XML conversion (Scenarios 3, 4 and 5). Making use of the XML structures and contextual cohesion in treaties, we are able to extract those "hidden titles" misplaced under `<p>` tags (Scenarios

3, 4 and 5). The worst scenario is 5 where the "authentic" title ("BASIS FOR CO-OPERATION") has been "concatenated" with its text during OCR (*Abbyy*) conversion; hence, it is impossible to disentangle those titles based on the XML structures. A good extraction strategy for *titled* and *untitled* articles is of great importance to our text categorization task (see Section 3.2). Our goal in the title extraction is (1) to extract all *titled* articles, (2) to extract all "authentically" untitled *articles* and (3) to retrieve as much as possible "formally" untitled articles and put them under the *titled* part of corpus.

```
<!-- Scenario 1: titled (nested titles) -->
<div type="part" num="1" title="GENERAL␣PROVISIONS">
<div type="title" num="I" title="OBJECTIVES,␣PRINCIPLES␣AND␣
    ACTORS">
<div type="chapter" num="1" title="Objectives␣and␣principles">
<div type="article" num="1" title="Objectives␣of␣the␣
    partnership">
<p>... Systematic account shall be ...</p></div>
<div type="article" num="2" title="Fundamental␣principles">
<p>L 317/7</p></div></div></div></div>

<!-- Scenario 2: titled -->
<div type="article" num="9" title="Essential␣Elements␣and␣
    Fundamental␣Element"><item num="1.">
<p> Cooperation shall be ...</p></item></div>

<!-- Scenario 3: "formally" untitled (title in <p> tag) -->
<!-- Example 1: -->
<div type="article" num="5"><p>EXPROPRIATION</p>
<p>I-Investments of investors of either Contracting Party in
    the territory of the other ...</p></div>

<!-- Example 2: -->
<div title="GENERAL␣PROVISIONS" num="I" type="chapter">
<p>A RTICLE 1</p><p>Objectives</p>
<item num="1."><p> The EFTA States and Lebanon shall establish
    a free trade area in accordance with ...</p></item></div>

<!-- Scenario 4: "formally" untitled (false positive in title
    attribute) -->
<div type="article" num="8" title="'"><p>Settlement of Disputes
     Between the Contracting Parties</p></div>

<!-- Scenario 5: "formally" untitled (concatenation of title
    with text) -->
<p>BASIS FOR CO-OPERATION The two parties undertake to
    strengthen ...<p>

<!-- Scenario 6: "authentically" untitled (short articles
    without titles) -->
<div num="One" type="section">
<p>The Participants will consider ways to enhance trade and
    investment between them.</p></div>
<div num="Two" type="section">
<p>Subject to their respective laws and regulations, ... </p></
    div>
```

Listing 3.3: Possible structures in XML where articles are stored

It is worth noting that as in scenario 1, nested structures of treaty elements are common in IIAs, where chapters, parts, sections, and articles are hierarchically organized. Some XML nested structures coincide with the treaty hierarchical structures, while other treaty hierarchical structures have been lost during conversion (PDF → *layout* XML → *content* XML). In Section 3.2 on the article title and texts extraction, the nesting structures in treaties are of special focus because we aim at increasing the recall of article extraction.

	category				
source format	**1**	**2**	**3**	**4**	**total (source)**
PDF	0	**537**	169	35	741
HTML	**1,518**	2	**486**	2	**2,008**
Microsoft Word	0	8	64	2	74
total (category)	**1,518**	547	719	39	**2,823**

Table 5: Source formats of the original files in the SNIS corpus

1: *EN_HTML_STRUCTURED*, 2: *EN_PDF_SEMI*,
3: *GOODMT_MIXED_SEMI*, 4: *BADMT_MIXED_SEMI*

If we observe the source formats and the four categories in a cross tabulation as in Table 5, we notice the uneven distribution of source formats in the various categories. Category 1 (53.8%) comprises only the English treaties converted from the HTML documents, whereas category 2 (19.4%) is mostly composed of the XML documents converted from the PDF documents in English. In category 3 (25.5%) there are more source files in HTML than those in PDF, and a large amount of the XML documents have been translated from French and Spanish with satisfactory quality. On the contrary, category 4 (1.3%) has the poorest language quality in English and consists mostly of the XML documents converted from the source files in PDF. Consequently, considerable attention must be paid when dealing with article extraction and analysis from treaties of various categories, because there are large discrepancies in the English language quality, the source format and the treaty structure in XML.

3.2 Extracting *Titled* and *Untitled* Articles

In order to perform article categorization, we first need to extract from titled articles their titles and the corresponding texts as training materials as well as untitled text blocks as test data. Preamble, conclusions, and annexes are stored in fixed XML tags; therefore, we do not take any texts from those three parts in our categorization

tasks. All titles and texts at the level of articles have been extracted from the `<body>` part in the *content* XML files. Despite the variety of article structure (be it nested or not) in XML (see Listing 3.3), we managed to develop strategies of extracting titles and texts as much as possible. Moreover, we not only need to extract the article titles but also their parent titles, because article titles within the same treaty can be duplicates if their parent titles are not taken into account. For instance, "Definitions" appeared 14 times in the treaty XML "{ASEAN,AUS,NZL}_2009-02-27" under 14 different chapters (e.g. "economic cooperation", "intellectual property", "electronic commerce", etc.). Without taking each chapter title to indicate which "Definitions" we are referring to, we will lose important snippets of article titles and texts. In the SNIS English corpus, long treaties such as treaties amongst countries and organizations tend to have article title duplicates across chapters. In total, 141 treaties contain article title duplicates, taking up 5% of the treaties in our corpus.

The strategy to store article title duplicates is to assign a new title by appending the parent title to the article title in the form of "article title | parent title". As we proceed along with title and text extraction, as long as title duplicates are still encountered during extraction, we take the parent title of the current title. Hence, multiple nesting of titles is tackled during extraction. Dictionaries in Python may be used to avoid redundant storing of duplicates which have been produced due to the replication of treaty contents. Take duplicates "Definitions" as an example: This title appeared twice in the treaty XML "{MEX,NLD}_1998-05-13" because the treaty was replicated twice in the XML file, and our extraction algorithm cannot detect any parent title in XML. The current title and its corresponding text will be written into a dictionary. The next time the extraction algorithm encounters the same title and text, the old entry of "Definition" in the dictionary will be replaced with the new one.

```
1 <item num="(B)">
2 <p> Contracting Parties or agencies authorized by the Contracting Party
      will exercise the rights of subrogation and enforce the claims of
     that investor and shall assume the obligations related to the
     investment.</p>
3 <p>Chapter 2:</p>
4 <p>DISPUTE RESOLUTION</p>
5 <p>Part 1:</p>
6 <p>SETTLEMENT OF DISPUTES BETWEEN AN INVESTOR OF THE SIGNING AND
     INVESTORS OF THE OTHER CONTRACTING PARTIES</p>
7 <p>Article 8.</p>
8 <p> Measures to resolve the dispute</p></item>
```

Listing 3.4: Mismatch of title nesting in *content* XML and treaty textual structure

Since category 1 has a strict mapping between XML structures and treaty structures,

the Python XML library `lxml`[17] can tackle the XML parsing easily. Even in cases of nested structure, retrieving the parent node of the current article title node is efficient. Categories 2, 3 and 4 have multiple possibilities of article structures due to the mismatch of the treaty element structure and the structure of *content* XML during conversion. The hierarchical treaty structures can be lost in XML, article titles might not be stored in `<div>` attribute *title* and the articles might be stored in `<p>` instead of `<div>` (see Listing 3.3). Another extreme case of mismatch is shown in Listing 3.4, where the chapter-part-article nesting has been placed under the `<item>` tag. Usually the tag `<item>` hosts bullet points of a certain treaty article. This makes the XML parsing with `lxml` a less appealing solution for the other three categories. The reason is that the parent and children nodes extracted by `lxml` do not necessarily correspond to the textual structure in the treaty. It could happen that after extraction, a chapter ends in the articles texts of the previous chapter. Hence, we can see the discrepancy between the treaty content structure and the XML structure can be large, which in turn influences our strategy of article extraction.

As a result, the strategy applied to categories 2, 3 and 4 is much complicated compared to that of category 1. Once the hierarchical nesting structure was lost during conversion from PDF to XML, the parsed segments by `lxml` could not assist relocating XML blocks to treaty structure while traversing the XML document. Using another Python library `BeautifulSoup` (aka `bs4`)[18] neatly tackles the structural mismatch by processing the XML elements sequentially. We can utilize the XML structural information, as well as the textual cues in treaty texts to retrieve the titles for "formally" untitled articles while separating the "authentically" untitled text blocks from the titled ones. The strategy used in other three categories is defined as follows:

1. We first use `bs4` to convert all `<div>` elements into dictionaries with attributes *title* and *type*, e.g. {`"num": "5", "type": "article", "title": u"Provision of Information"`}.

2. Then we convert `<p>` elements to strings and store the dictionaries converted from `<div>` and the strings sequentially into a list. Hence, we preserve the XML structural information and the textual structure of treaty texts.

3. Then we look for textual cues for articles, parts, sections and chapters with the help of regular expressions (e.g. "Article", "chapter", etc.). The latter three levels are handled in parallel with article structure, otherwise, we will

[17] http://lxml.de/ (accessed 11 Jan 2017).

[18] https://www.crummy.com/software/BeautifulSoup/bs4/doc/ (accessed 04 April 2017).

lose hierarchical information of nested titles.

4. After we anchor the textual cues for titles, we examine the text pattern and decide whether to take the string behind the cue words as the title or the next element in the list as title, if it is capitalized and is shorter than the longest retrieved title in category 1 (*EN_HTML_STRUCTURED*).

When extracting the titles and texts, the following preprocessing steps have been undertaken to remove undesired noise in the experiments later on:

- Clean the titles: OCR error correction (rule-based)
  "A rticle 1.2: G eneral D efinitions" → "Article 1.2: General Definitions" by defining the rule: If a single character in upper case is followed by more than one characters in lower case, join them.

- Normalize the titles: Roman numerals (e.g. "IV") and English numeral words (e.g. "four") to Arabic numerals (e.g. "4")
  Please note that it is only for the ease and simplicity of processing later on, we convert the numerals to a unified format. However, in corpus linguistics, it is encouraged to preserve the original forms and add special markups of the unified form of the normalized numeral.

- Clean the titles and texts: Filter out non-English words with the Python `PyEnchant`[19] library
  Due to the imperfection of SMT translation, we end up having many non-English words in titles and texts. In order to apply standard English NLP tools (e.g. lemmatizer) and facilitate feature engineering in machine learning, filtering out the foreign words is a realistic solution.

The extraction and processing of articles have implemented in Script (S1).

Manual evaluations have been implemented to guarantee the quality of article titles because quite often false positives (e.g. punctuation, numerals and article texts) have ended up in the `<div>` attribute *title*. Punctuation and numerals are stripped from the titles. Except for the stop word "the", if a title begins with stop words (from the Natural Language Toolkit (`NLTK`) [Loper and Bird, 2002] stop word list for English), it would not be considered as a *titled* article. This title and its corresponding texts are then concatenated and stored as an *untitled* article. For instance, "if one of the contracting parties" was put under the *title* attribute in XML files. With our simple techniques mentioned earlier, we can remove this string from the pool of titles.

[19]http://pythonhosted.org/pyenchant/ (accessed 13 March 2017).

After the extraction and preprocessing of titles and texts, we stored them together with treaty name, category, source format, and source language into Python dictionaries (examples shown in Listing 3.5). The tokens and types for *titled* and *untitled* articles are shown in Table 6. We observe that while the number of titled articles is three times that of untitled articles, titled and untitled articles do not share a large portion of the vocabularies. The total number of types in the corpus is almost equal to the sum of vocabularies from the titled and untitled articles. In total, we have extracted 10,074 untitled and 34,524 titled articles with more than nine million tokens to experiment. Interestingly, we found out that untitled and titled articles are mixed in 453 treaties (16% of the SNIS corpus).

```
Item format: ((treaty, title/generic counter, category, source format,
    source language), texts)

Untitled: ((‘{GCC,USA}_2012-09-25.xml’, 2, ‘cat2’, ‘ABBYY’, ‘en’), ‘The
     parties shall consider possible ...’)

Titled: ((‘{MYS,URY}_1995-08-09.xml’,‘promotion and protection of
    investments’, ‘cat3’, ‘html’, ‘es’), ‘each Contraction Party shall
    promote ...’)
```

Listing 3.5: Items in Python dictionaries for titled and untitled articles

	article	token	type
untitled	10,074	1,618,395	40,506
titled	34,524	7,505,258	42,636
total	44,598	9,123,653	73,134

Table 6: Token and type counts for titled and untitled articles

category	1	2	3	4	total
untitled	4,226	3,424	2,350	74	10,074
titled	16,532	12,014	5,899	79	34,524
total	20,758	15,438	8,249	153	44,598

Table 7: Cross tabulation of four categories and titled and untitled articles

1: *EN_HTML_STRUCTURED*, 2: *EN_PDF_SEMI*,
3: *GOODMT_MIXED_SEMI*, 4: *BADMT_MIXED_SEMI*

The average length of titled texts is 217.39 tokens, the untitled 160.66 tokens. It can be seen that the length discrepancy between titled and untitled parts is not as large as expected since one might have the impression that untitled articles are much shorter than titled articles. The average length of article titles is 3.84 tokens.

As shown in Table 7, the translated texts in the untitled and titled parts are 23% and 17%, respectively.

The quality of the extraction of titles and texts has been manually evaluated by randomly selecting three documents from categories 1, 2 and 3, as XML files in category 4 are of unsatisfactory quality in English translation and are not suitable for further analysis. Treaty XML documents "{ALB,CHN}_1993-02-13" (category 1, 12 *untitled* articles), "{ALB,LTU}_2007-03-28" (category 2, 13 *titled* articles, 1 *untitled* article) and "{ARE,SYR}_1997-11-26" (category 3, 3 *untitled* and *titled* articles, respectively) have been chosen to evaluate precision and recall in the extraction. The first two documents were originally in English, whereas the third treaty was translated from Arabic into English, with about 5% of foreign words remaining in texts. The precision and recall of titled and untitled articles in three XML documents have reached 100%. The evaluation led us to conclude that the two distinctive extracting strategies for four categories have performed well to retrieve as exactly and as much as possible.

We have found out that our extraction algorithms have skipped 49 treaties and extracted articles from 2,774 treaties due to the following reasons:

1. 45 out of 49 treaties were in PDF documents, which has induced OCR errors during conversion.

2. 73.5% (36 out of 49 skipped treaties) treaties were converted from PDF documents in the source languages such as Arabic (55%), Slovak, Romanian, Kazakh, Kirghiz and then translated into English. Hence, the language quality is extremely poor because of OCR errors and the difficulties in translating from those source languages into English, with mostly non-English words in texts. Our extraction algorithm keeps only the articles with English words. Most of the treaties under category 4 (*BADMT_MIXED_SEMI*) have been ignored.

3. Cue word such as "paragraph" was not included in the extraction algorithm we defined (four treaties with "paragraph" indicating provisions), as they appear extremely seldom.

We observe from 34,524 article titles that they differ from each other in word forms (singular, plural), in format (upper case, lower case), in stop words (with, without) and in word order, etc. In order to gain a more condensed representation of distinctive titles, we lemmatized and lowercased titles, removed stop words and sorted words in titles in the alphabetical orders. This renders the list of 35,524 retrieved titles in 5,101 unique normalized forms. Table 8 lists three examples of unique nor-

unique normalized titles	frequency
“admission”, “investment”	28
“contracting”, “dispute”, “settlement”, “state”	38
“compensation”, “dispossession”	72

Table 8: Unique normalized titles after preprocessing

frequency range	count
>=1000	5
>=500, <1000	6
>=100, <500	38
>=50, <100	29
>1, <50	1,691
1	3,332
total	**5101**

Table 9: Frequency distribution of unique normalized titles after preprocessing

malized titles and their frequency. The frequency distribution of normalized titles shown in Table 9 is extremely uneven: We have 65.3% of titles which appear only once; 33.2% of the titles appear more than once yet less than 50 times; only eleven normal titles have been used very frequently, i.e. more than 500 times. As a result, it would be challenging to categorize articles based on an uneven frequency distribution. In the next chapter, we discuss our choice of methods suitable to our data structure thoroughly.

4 Methods, Tools and Experiments

The aim of the thesis is to assign titles to untitled articles based on the pre-existing knowledge in the titled corpus. In order to provide a clearer picture of the pipeline of text categorization, we present four important steps (① ② ③ ④) as illustrated in Figure 15.

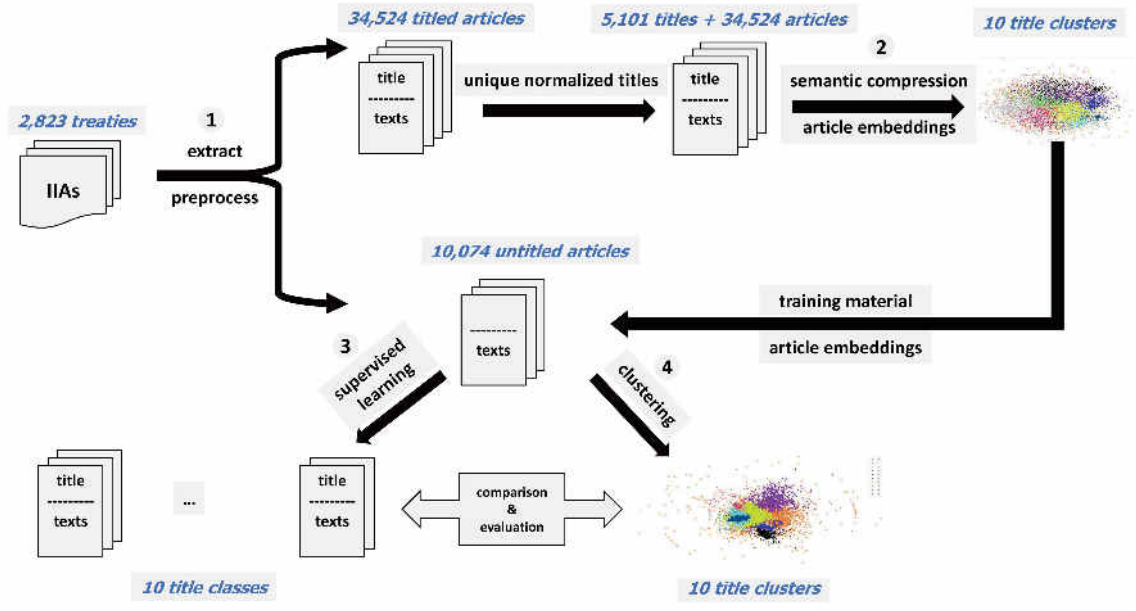


Figure 15: Pipeline of text categorization

We have discussed in Section 3.2 how to extract and process titles and texts on the level of treaty article. This is ***step*** ①, extraction and preprocessing, after which we obtained 5,101 unique normalized titles and 34,524 article texts. As it is not feasible to use around 5,000 categories in a categorization task, we first need to compress the titles into more condensed, meaningful categories and then use those in our text categorization task.

Consequently, we utilize word and document semantics as well as document clustering to compress the similar articles (***step*** ②). On the one hand, articles in IIAs are interrelated, yet are different from one another in theme, for instance, some focusing on dispute settlement, others addressing issues in monetary transfer. On

the other hand, the articles all use standardized legal language, and they are put together within the same treaty in a logical order, so that, taken together, they form coherent legal documents. Moreover, as we have a relatively small corpus (with nine million tokens), we can benefit substantially from word embedding expansion (see Section 2.3.3.2). Hence, we adopt embeddings as features in document clustering. We also employ a pre-existing taxonomy of IIA topics to supervise clustering as proposed in Aggarwal et al. [2004] partially. As a result, we are able to cluster 34,524 treaty articles into ten categories in ***step*** (2).

In ***step*** (3) and ***step*** (4), we perform text categorization using supervised learning methods and clustering methods. To ascertain the efficacy of word embeddings in text categorization by expanding the meaning of words, we also compare the accuracy of supervised learning and partially supervised clustering during testing (aka assigning topics to untitled articles).

Experiments for machine learning were carried out with the Python libraries `scikit-learn` [Pedregosa et al., 2011] and `TensorFlow`[1].

4.1 The Pipeline of Treaty Article Categorization

In this section, we pinpoint the fundamental methods in each step of the pipeline.

(1) Extracting and Preprocessing of treaty titles and texts (Scripts (S1), (S2))

   a) Parsing XML

   b) Extraction of titles and texts based on the XML document logical structure and the textual structure of IIAs

   c) Titles and texts: lemmatization with `TreeTagger`[2] [Schmid, 1994]

   d) Titles: stop word removal, numeral normalization, foreign words filtered, tokens sorted in alphabetical order

   e) Texts: numeral normalization, foreign words filtered

(2) Semantic article compression and clustering of the titled corpus (Scripts (S3), (S4), (S5), (S6))
   Clustering with a given number of clusters is ideal with k-means algorithm. The idea is to use the pre-defined definitions of ten topics in IIAs (see Section

[1] https://www.tensorflow.org/api_docs/python/ (accessed 20 April 2017).

[2] http://www.cis.uni-muenchen.de/~schmid/tools/TreeTagger/ (accessed 15 April 2017).

4.3.1) as the initialized centroids around which the titled articles can group. Each titled article is represented by document embeddings which are composed by the pretrained word embeddings from *Google News*. We carry out experiments on how to deduce the suitable document embeddings and retrain word embeddings with our SNIS corpus. The results of k-means clustering are labeled data in which each titled article has a class label assigned through clustering. We then utilize the transformed titled corpus to train and tune our classifiers in the supervised learning settings.

(3) Assigning the titles to untitled articles using supervised learning (Scripts (S7), (S8))
We train and tune six classifiers, i.e. KNN, linear and non-linear SVM, MLP, SGD as well as CNN.

(4) Assigning the titles to untitled articles using k-means clustering (Script (S6))
We use the retrained word embeddings from the SNIS corpus in Step (2) to generate document embeddings in the untitled corpus. Then we cluster the untitled articles and compare the results with those in supervised learning.

The following sections are devoted to describing tools and settings in each step of the processing pipeline.

4.2 Word and Article Embeddings in the SNIS Corpus

As we have a relatively small corpus, expansion of corpus-customized word embeddings based on pretrained *Google News* embeddings is the key to success in clustering and classification tasks. For the titled corpus, we first generate the title embeddings and text embeddings separately and then construct document embeddings for articles through vector composition. Mitchell and Lapata [2008] provide us inspirations on how to assign weights to titles and texts in additive and multiplicative functions.

To start with, we test different setups for generating document embedding representations for article titles with the Python libraries `doc2vec` and `word2vec`[3], in order to select the desired method that best fits the domain of our SNIS corpus.

We test and compare the outputs of title embeddings that are generated using the following setups:

- `DBOW` with `doc2vec` library

[3]https://github.com/jhlau/doc2vec (accessed 01 May 2017).

As reported in Lau and Baldwin [2016], the settings of `DBOW`, where the order of words in the document is ignored, work better than `DMPV` settings, where the paragraph is regarded as a token in the input layer, and the sequence of words is considered. We use the pretrained `doc2vec` models on English *Wikipedia* and *Associated Press News* to generate representations of article titles[4].

- TF/IDF-averaged word embeddings to represent the article with the pretrained *Google News* word embeddings.
- Averaged word embeddings to represent the article with pretrained *Google News* word embeddings.

After selecting the best strategy from three options to represent article titles and article texts, we need to compute the TF/IDF score of the vocabulary and retrain the word embedding with our corpus.

For computing the TF/IDF score of the vocabulary and retraining word embeddings with the SNIS corpus, we only use the titled corpus. If a word does not exist in the *Google News* word embedding (quite rare), we simply ignore that word when computing document embedding. TF/IDF scores are computed with the `scikit-learn` class `TfidfVectorizer()`.

For retraining word embeddings on the SNIS corpus, we use the Python `gensim`[5] library provided by Lau and Baldwin [2016] and the pretrained *Google News* embeddings. The hyperparameters we tune in training are `vector_size, window_size, min_count, sampling_threshold, negative_size, train_epoch, paragraph_vector` and `worker_count`. Subsampling threshold in `word2vec` helps downsample frequent words; negative sampling means to randomly select a small set of co-occurrences instead of sampling all the co-occurrences in the corpus [Mikolov et al., 2013b, 3-4].

After obtaining the retrained word embeddings tailored to the SNIS corpus, we need to compose the document embeddings for titled articles through additive and multiplicative functions applied to title embeddings and text embeddings. Various weighted addition possibilities (i.e. finding optimal weights α and β in $v_{article} = \alpha v_{title} + \beta v_{text}$) have been tested and we evaluated the efficacy of vector composition together with the clustering quality (through the Silhouette coefficient, topic modeling and visualization, see Section 5.2).

For articles that share the same normalized title after preprocessing, the variability

[4]Available at https://github.com/jhlau/doc2vec (accessed 01 May 2017).

[5]https://radimrehurek.com/gensim/models/doc2vec.html (accessed 10 May 2017).

of texts is high. For a condense representation of texts sharing the identical normalized title, we took the average article embeddings for those articles. In the end, we obtained 5,101 article embeddings which we could use as features in k-means clustering.

4.3 Partially Supervised Clustering of Articles Using Main Topics in IIAs

As we have discussed in Section 2.5.1 on k-means clustering and Section 2.6 on partially supervised clustering, the initialization of clustering can greatly influence the quality of cluster partitioning. Recall we have generated document embeddings for 5,101 unique normal titles and their corresponding texts in the titled corpus. Hence, we discuss in this section how to conduct semantic compression of title types by using the pre-existing definitions of main topics in IIAs, in order for a better representation of the label set we can use for article categorization later.

4.3.1 Topics in IIAs

Ten main topics of IIAs are commonly agreed upon to form an exhaustive list of topics an investment treaty can cover, as summarized in Salacuse [2015, 141-150].

1. Treaty title and statement of purpose
2. Scope of application of investment treaties
3. Conditions for the entry of foreign investment and investors
4. General standards of treatment of foreign investments and investors
5. Monetary transfers
6. Expropriation and dispossession
7. Operational and other conditions
8. Losses from armed conflict or international disorder
9. Treaty exceptions, modifications, terminations
10. Dispute settlement

On the other hand, the *IIA Mapping* Project[6] from UNCTAD provides a similar list of nine topics in IIAs. The list is organized in a hierarchical taxonomy, e.g. with the subcategories such as "national treatment" and "most-favored-nation treatment" for the topic "standards of treatment"[7]. The content mappings summarized in Salacuse [2015] and proposed by UNCTAD share the same paradigm, with the latter using more fine-grained categories and a hierarchical structure. We did not adopt the UNCTAD hierarchical taxonomy for this master thesis, due to the following two reasons: (1) We firstly are in need of a general understanding of IIA topics which non-hierarchical categorization can partition the dataset in an even way because the relation between clusters is often undetermined (see Manning and Schütze [2000, 498]); (2) Hierarchical classification is more demanding in computation and requires a better understanding of nesting of topics amongst one another.

The first topic from Salacuse [2015] should not be included in our study because it refers to the treaty title and preamble. This leaves us with nine topic domains. Since k-means clustering works for general purposes of classification (esp. with a small number of clusters) and it requires a given number of clusters as input (see Section 2.5.1), we opted for this clustering technique in the thesis to "compress" the heterogeneous set of titled articles. It remains to be tested during the clustering process whether nine is the final number of clusters.

4.3.2 Partially Supervised Clustering

In this thesis, we aim at using the weak supervision in document clustering, as Aggarwal et al. [2004] explained in their experiment with unlabeled data. They termed the approach of using a priori knowledge as "centroids" *partially supervised clustering*, one type of semi-supervised learning (see Section 2.6). In our experiment, we possess 5,000 unique titles after normalization. It is infeasible to use them as our classes when labeling the untitled articles because we cannot generalize our learned knowledge with 5,000 classes. That being said, we need to make use of document clustering to "compress" this list of normalized titles. Luckily we can set the underlying topics of IIAs introduced in Section 4.3.1 as the "centers" for the future clusters. This is called the initialization of clusters. Other factors we need to consider in clustering are features, the number of clusters, iteration passes, etc. We

[6] A collaborative initiative to provide a detailed analysis of over 2,500 investment agreements based on 100 options for treaty design, see http://unctad.org/en/pages/DIAE/International%20Investment%20Agreements%20(IIA)/IIA-Tools.aspx (accessed 20 May 2017).

[7] http://investmentpolicyhub.unctad.org/IIA/mappedContent (accessed 20 May 2017).

can either use lexical, distributional or embedding features in order to capture the similarity and dissimilarity of documents. In the case of having a relatively small corpus, we should opt for the features that encode rich semantics of documents. Certainly, the latter two feature engineering approaches are more appropriate for our task.

The product of partially supervised clustering is the "labeled" training material, where each titled article is mapped to a cluster (one topic of IIAs). We can then generalize the knowledge we learn from the training instances and make predictions of the topic given an untitled article.

4.3.3 Evaluation of K-means Clustering

Evaluation of k-means clustering is to decide the best partitioning method of the dataset. In relation to this, we discover the number of clusters with the help of our prior knowledge of IIA topics. In this thesis, we combine three methods to evaluate the results of clustering: (1) the Silhouette coefficient which measures the distance between the mean of instances in the same cluster (i.e. intra-cluster) and the mean of instances from the nearest cluster, (2) visualization with MDS as well as (3) topic keywords generated by topic modeling using LDA.

MDS is applied to project a high-dimensional representation of data into a low-dimensional space and to analyze similarity or dissimilarity of data as "distances in a geometric space"[8]. The metric used to compute dissimilarity or similarity can be cosine similarity. To visualize our k-means clustering, we use the following dissimilarity measure: distance = 1 - cosine similarity.

Topic modeling is applied to each cluster of documents to generate 20 representative key words for that cluster. The `gensim` library was used[9] to implement LDA models. Tunable hyperparameters are 100 passes (`passes`) over the supplied corpus, updating model five times (`update_every`) every 100 documents (`chunksize`).

As we know, k-means clustering is quite sensitive to how we initialize the centroids; hence, we managed to make use of pre-existing definitions in Salacuse [2015] for topics in IIAs. The texts for definitions are mainly taken from Chapter 5 in Salacuse [2015, 141-154][10]. Definitions for each topic were listed in Appendix C, whose

[8] http://scikit-learn.org/stable/modules/manifold.html#multidimensional-scaling (accessed 10 May 2017).

[9] https://radimrehurek.com/gensim/models/ldamodel.html (accessed 10 May 2017).

[10] As there has not been any freely available digital copy of the book, we managed to firstly scan Salacuse [2015, Chapter 5: The General Structure of Investment Treaties, 141-154],

embeddings were also computed using the best settings described in Section 4.2. For an unseen word in definitions, its TF/IDF score was computed using add-one smoothing. The iteration passes, 100 and 200, were tested in all runs of k-means clustering.

The titled articles together with their assigned cluster membership would then be used as training and tuning material for text classification. As we do not have access to the true label for each titled article, we simply assume that the assigned cluster membership can act as a proxy for the true label. This is also why we call this approach *partially supervised clustering.* One general problem of clustering techniques is that we usually do not have access to gold standards where the true labels are correctly assigned to their instances unless we manage to generate human annotations for the testing instances. Fortunately, we have obtained the 100 gold labels for 100 titled articles (Annotation (A1)), thanks to the great support by Prof. Dr. Peter Egger, an expert in IIAs and international trade. The expert was provided with the article titles and their corresponding texts, for which he chose one label from the given set of labels, i.e. the topics of the resulting clusters. We report the accuracy of clustering later on in Section 5.2. Bear in mind that the quality of clustering can influence the performance of classifiers in text classification.

4.4 Assigning Topics to *Untitled* Articles: Classification

We applied six different classifiers (KNN, linear SVM, non-linear SVM, MLP, SGD and CNN) to our training and tuning sets generated by k-means clustering in Section 4.3. The first five classifiers were trained using `scikit-learn`, the last classifier,

and then converted the images to texts in *Microsoft Word* using an online OCR platform: https://www.onlineocr.net/ (accessed 10 May 2017). This platform offers free service of OCR conversion (15 images per hour) upon registration. Sentences in the definitions of topics are mostly literally selected from the chapter. As certain topics are only briefly discussed in Chapter 5, we also consulted the other chapters to generate comprehensive definitions for the topics listed in Appendix C. For topic 0, additional informative sentences have been taken from ibid., Chapter 8: Investment Promotion, Admission, and Establishment, 8.1 State Sovereignty and Foreign Investment, 213-214. The definition of topic 2 was created by ourselves based the results of clustering, see *Number of clusters* in Section 5.2. Some sentences in topic 4 are taken from ibid., Chapter 14: Investment Treaty Exceptions, Modifications, and Terminations, 14.1 The Tensions of Investment Treaties, 376. The definition of "losses from armed conflict or internal disorder" under topic 8 has been taken from ibid., Chapter 13: Other Treatment Standards, 13.4 Compensation of Losses Due to War, Revolution and Civil Disturbance, 367-368. The definition of topic 9 was taken from ibid., Chapter 1: A Global Regime for Investment, 1.4 The Application of Regime Theory to Investment Treaties, 10. We manually proved the cohesion and coherence of sentences and made only minimal changes to connectives and determiners, to create internally coherent definitions of each topic.

CNN, was trained with the `TensorFlow` library. For each classifier in `scikit-learn` we use `GridSearchCV()`[11], which performs exhaustive search over specified parameter values for a classifier. The parameters of the classifiers are optimized by cross-validated grid search over combinations of parameters. For instance, we assign two parameters to a classifier, one parameter with two possible values, the other with three. The combination of parameters results in six various combinations in the grid search. After the grid search, we can output the best set of parameters and the best score (e.g. accuracy) with that set of parameters, as well as use this setting of parameters to predict new instances. We used 5-fold cross-validation in our grid search.

In the following, we listed the parameter we used for each classifier. Features are the 500 most frequent vocabulary items in the BoW model, transformed by TF/IDF scores, lowercased, with stop words (from the `NLTK` English stop word list) filtered. Please note that the explanations of parameters and values are taken from `scikit-learn` manuals on each classifier; the links are documented in the footnotes for each classifier.

1. `KNeighborsClassifier()`[12]
   Parameters and values[13] in the cross-validated grid search:
   `"weights":["uniform","distance"],`
   `"algorithm":["auto","ball_tree","kd_tree","brute"],`
   `"n_neighbors":[5,10,15].`

   - `"weights"`: weight function used in prediction.
     `"uniform"`: all points in each neighborhood are weighted equally;
     `"distance"`: closer neighbors of a query point will have a greater influence than neighbors which are further away.

   - `"algorithm"`: algorithm used to compute the nearest neighbors.
     `"auto"`: attempt to decide the most appropriate algorithm based on the values passed to training methods; `"brute"`: brute-force approach (aka proof by exhaustion); `"kd_tree"`: a binary tree structure which recursively partitions the parameter space along the data axes, dividing it into nested orthotopic regions into which data points are filed; `"ball_tree"`:

[11] http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html (accessed 10 May 2017).

[12] The listing of parameters and values for the KNN classifier is mainly taken from http://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html (accessed 10 May 2017).

[13] Format: `"parameter":[value1,value2,...]`.

ball trees partition data in a series of nesting hyper-spheres[14].

- `"n_neighbors"`: the number of neighbors to use.

2. `SVC()`[15] and `LinearSVC()`[16]
   `SVC()` implements the OvO approach for multiclass classification, whereas `LinearSVC()` implements OvR multiclass strategy. Parameters and values in the cross-validated grid search of `SVC()`:
   `"C":[1,10,100,1000],`
   `"kernel":["linear","rbf"],`
   `"gamma":[0.001,0.0001]` (`LinearSVC()` is trained with the same set of parameters except `"kernel"`.).

   - `"C"`: penalty parameter C of the error term.
   - `"kernel"`: specifying the kernel type to be used in the algorithm. `"linear"`: transformation function of input x and non-kernel counterpart y, $k(x, y) = x^T y + C$; `"rbf"`: aka Gaussian kernel, $k(x, y) = exp(\sigma||x - y||^2)$[17].
   - `"gamma"`: kernel coefficient for `"rbf"`.

3. `MLPClassifier()`[18]
   Parameters and values in the cross-validated grid search:
   `"hidden_layer_sizes":[(100,),(50,)],`
   `"activation":["logistic","tanh","relu"],`
   `"alpha":[1.0e-03,1.0e-04,1.0e-05].`

   - `"hidden_layer_sizes"`: the number of neurons in the hidden layer.
   - `"activation"`: activation function for the hidden layer (see Section 2.4.4).
   - `"alpha"`: L2 penalty (regularization term) parameter, to penalize extreme

[14]Explanations on `"kd_tree"` and `"ball_tree"` are taken from http://scikit-learn.org/stable/modules/neighbors.html (accessed 10 May 2017).

[15]The listing of parameters and values for the SVM classifiers is mainly taken from http://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html (accessed 10 May 2017).

[16]The listing of parameters and values for the linear SVM classifier is mainly taken from http://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html (accessed 10 May 2017).

[17]For more on kernel functions, see http://crsouza.com/2010/03/17/kernel-functions-for-machine-learning-applications/#linear (accessed 10 May 2017).

[18]The listing of parameters and values for the MLP classifier is mainly taken from http://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html (accessed 10 May 2017).

parameter weights [Raschka, 2015, 66].

4. `SGDClassifier()`[19]
   `SGDClassifier` supports multiclass classification by combining multiple binary classifiers in an OvR scheme. Parameters and values in the cross-validated grid search: `"loss":["hinge","log"],`
   `"penalty":["l1","l2"],`
   `"alpha":[1.0e-01,1.0e-02,1.0e-03,1.0e-04,1.0e-05]`.

   - `"loss"`: loss function.
     `"hinge"` gives a linear SVM; `"log"` loss gives logistic regression, a probabilistic classifier.
   - `"penalty"`: the penalty (aka regularization term) to be used.
     `"l1"`: feature selection; `"l2"`: tackling overfitting.
   - `"alpha"`: constant that multiplies the regularization term.

The CNN classifier used in this thesis replicates the CNN architecture described in Kim [2014] and Zhang and Wallace [2015], on which a *GitHub* project `cnn-text-classification-tf`[19] was published. We used the source code from the project on *GitHub* and tried to configure the best hyperparameter setting for our classification task. Besides, to combat overfitting in the model, we used dropout to assist feature selection. Hyperparameter tuning settings are as shown in Table 10.

In this table, "embedding" is the dimensionality of our word embeddings. "Filter sizes" denotes the number of words we want our convolutional filters to cover. "No. filters" means the number of filters per filter size[20]. For example, [3, 4, 5] indicates that we use the filters to slide over three, four and five words respectively, for a total of $3 \times no.filters$. Stride in CNN is defined as one for filtering, meaning each feature window will move consecutively further in scanning the input matrix. "Batch size" is defined due to vectorization, implying at each step of training, the network will take 64 instances for training. "Epoch" describes the number of passes over the whole training set, i.e. how many times will the network update its weights on the training set. "Dropout" is a technique of pruning, i.e. the network "disables" a fraction of its neurons randomly during training; however, in testing, no neurons should be disabled (dropout is not applied to testing).

[19]The listing of parameters and values for the SGD classifier is mainly taken from http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.SGDClassifier.html (accessed 10 May 2017).

[19]https://github.com/dennybritz/cnn-text-classification-tf (accessed 05 Jan 2017).

[20]http://www.wildml.com/2015/12/implementing-a-cnn-for-text-classification-in-tensorflow/ (accessed 10 April 2017).

The goal of implementing CNN in text classification task is to test the efficacy of deep neural network in text classification. Because each run took 7-8 hours to finalize, we did not tune our CNN classifier with many different hyperparameter settings. Randomly initialized embeddings were used; each run performed a 10-fold cross-validation.

	embedding	**filter size**	**no. filters**	**dropout**	**batch size**	**epochs**
run1	50	**5,5,5**	20	0.1	64	10
run2	50	**3,4,5**	20	0.1	64	10

Table 10: Hyperparameter settings in CNN

The lowercased training and tuning material was fed into the classifiers in the lemma forms. The test set (10,074 untitled articles) was the untitled SNIS corpus. As we were in need of a human-generated gold standard for evaluation, Prof. Dr. Peter Egger also annotated 100 test instances (aka assigned topic labels to untitled articles, Annotation (A2)) which we then used to evaluate the performance of our classifiers (see Section 5.3).

4.5 Assigning Topics to *Untitled* Articles: Partially Supervised Clustering

As a comparison with assigning titles with supervised methods, we also attempted to cluster our test instances with retrained word embedding using the SNIS corpus. We believe that this comparison would assist us to gain insights into the efficacy of word and document embeddings, as well as the advantages and disadvantages of supervised and unsupervised methods. The untitled article embeddings were generated using the best setup based on the results from clustering titled corpus. K-means clustering for the untitled corpus works the same as described in Section 4.3. Also, TF/IDF scores for unseen words in untitled articles were smoothed with add-one smoothing as suggested in Sarkar [2016, 182]. It is hard to determine without the comparison on the same evaluation set, which method (classification or clustering) on the untitled part would work better. Supervised methods highly depend on the accuracy of partially supervised clustering and have the advantage of learning from concrete mappings between the features and the class labels. In contrast, the only magic unsupervised clustering can lean on is the retrained word embeddings of the SNIS corpus. It remains to be observed how well do the learned word embeddings generalize in the unseen data, remains to be observed.

5 Text Categorization: Results and Evaluation

In this chapter, the results of experiments in the pipeline of text categorization are presented. We first discuss our results on choosing the best strategy for document embeddings (here a document being an article title or an article text). Then we report the parameters to retrain word embeddings in the SNIS corpus and presented how we generate article embeddings based on title and text embeddings. We also discuss the results of partially supervised clustering of the titled articles using pretrained and retrained word embeddings. Subsequently, we presented the results of mapping the titled articles to a given topic taxonomy in IIAs (ten topics) using k-means clustering. We hence can use the cluster labels and the titled corpus to train and tune our supervised classifiers and make predictions on the labels of the untitled articles. In order to explore the efficacy of word embeddings as features in machine learning tasks, we clustered the untitled titles into ten categories with merely the retrained word embeddings from the SNIS corpus as features. Last but not least, article classification and article clustering of the untitled SNIS corpus are compared regarding the overall accuracy and the individual accuracy in each category. The findings presented in this chapter demonstrate the potential for partially supervised clustering in categorizing treaty articles, as well as the efficacy of embeddings as features in machine learning.

5.1 Article Embeddings: Title and Text

In order to prove the efficacy of word embeddings on representing documents, we compared the cosine similarity of 100 randomly selected titles with vectors computed by averaging word embeddings (*avg_w2v*), TF/IDF-weighted averaging word embeddings (*tfidfavg_w2v*), pretrained document embeddings using DBOW on *Associated Press* news (*d2v_apnews*) and pretrained document embeddings using DBOW on English *Wikipedia* (*d2v_enwiki*).

		cosine metric				
		1	**2**	**3**	**4**	
title1	**title2**	avg_w2v	tfidfavg_w2v	d2v_apnews	d2v_enwiki	avg(1,2)
access, market	access, establishment, market	0.88	0.87	0.977	0.99	0.87
access, market	access, court, tribunal	0.36	0.517	0.937	0.97	0.43

Table 11: Cosine similarity scores for the vectors computed by `word2vec` and `doc2vec` with the pretrained *Google News* word embeddings

We computed the title vectors with titles preprocessed as described in Section 3.2 (lowercased, lemmatized, stop words removed, sorted alphabetically) and two examples are shown in Table 11. Each title is represented as a list of sorted content words. It is a clear-cut case for human judgment that the first pair is a similar pair, while the second pair is clearly a dissimilar pair. We expected the cosine similarity of our chosen document representations to be discriminative between the similar and dissimilar pairs. The `doc2vec` scores were not informative in dissimilarity because the cosine metrics 3 and 4 for similar and dissimilar pairs were both high. On the contrary, there was positive evidence for the cosine scores of title vectors computed by `word2vec`. The `doc2vec` document embeddings output very high scores (around 0.90 on average) even for quite dissimilar pairs such as "investment, promotion, protection" and "body, corporate, govern, management, operation, senior" (0.39, 0.31, 0.99, 0.99 for cosine metrics 1, 2, 3, 4). As suggested by Lau and Baldwin [2016] (see Section 2.3.3.2), the effect of `doc2vec` can diminish for shorter texts (13 tokens per sentence); however, we cannot find any significant change in the cosine similarity scores even for longer sentences from our corpus. As a result, we decided to use word embeddings computed by `word2vec` to compose document representations for titles and texts.

Furthermore, in order to evaluate the measurement of cosine metrics of `word2vec` generated vectors, we compared the automatic output against human annotations generated on our own. As we could not decide at this stage, without the performance of partially supervised clustering, whether *avg_w2v* or *tfidfavg_w2v* is a better fit for our learning problem, we took the average score of cosine metric 1 and 2 as shown in the last column of Table 11. We took the minimum averaged score of *avg_w2v* and *tfidfavg_w2v* of the similar titles (e.g. 0.65) as the cutoff to determine whether two titles are semantically similar. Out of 100 randomly chosen pairs, 0.65 is the cutoff for a similar pair of titles; any title pair that scored less than 0.65 was regarded as a dissimilar title pair. Then we compared the human judgment with the automatically

computed cosine values; the agreement rate is 86% for 100 randomly chosen pairs (among which 34% are similar pairs). In other words, the averaged word embedding scores have a tendency to predict the paired titles as similar, as the precision of similar pairs was extremely high (32 out of 34 correctly predicted).

So far, we can identify that the word embeddings computed by the `word2vec` paradigm, compared to that computed by `doc2vec`, can better represent texts and model the similarity and dissimilarity in the domain of IIAs. When it comes to which vector composition method (averaging the word vectors or TF/IDF-weighted averaging the word vectors) is better for our learning problem, it remains to be examined together with the results of article clustering, the visualization of clustering and the representative keywords for each topic in the next section.

An interesting by-product we discovered during the annotation of article titles is that there could actually be errors in the original treaty titles. Article 5 in the treaty XML "{BLEU,LKA}_1982-04-05" has the title "Exportation" with its corresponding text: "Neither of the parties shall take measures of expropriation, nationalisation or dispossession, or any other measures having effect equivalent to expropriation, nationalisation or dispossession, against investments belonging to nationals or companies of the other Contracting Party, unless such measures are taken in the public interest, on a non-discriminatory basis, and under due process of law, and provided that provision be made for prompt, effective and adequate compensation." The correct title to this text would probably be "Expropriation", as "exportation" means "the sending out (of commodities) from one country to another" according to the *Oxford English Dictionary*[1]. It is worth noting that this treaty XML has been converted from a PDF document with the blurred texts generated by typewriter. We assumed that all the titles and texts that come with the titled corpus should be correctly mapped; therefore, such errors must be really rare. Nonetheless, this error certainly confirms our expectation at the outset that just looking at the titles or the texts for titled articles is not sufficient to represent the article meaning. Consequently, we construct the document representations for articles by weighted vector composition of the title vectors and the text vectors as Mitchell and Lapata [2008] proposed. The best weighting strategy of title and text is evaluated in parallel with the clustering results of articles in the next section.

We now discuss the parameters we set for the retraining of word embeddings with our SNIS corpus. With the titled part of the corpus (titles and articles all used, lemmatized, lowercased), we loaded the pretrained *Google News* embeddings as the

[1] http://www.oed.com/view/Entry/66701?redirectedFrom=exportation#eid (accessed 10 May 2017).

initialized weights for input words in the Python class `gensim.models.Word2Vec()`. As *CBOW* runs faster than *skip-gram* and is not in need of a large amount of training material, we used the default training algorithm, CBOW. Dimensionality of the vectors has been kept 300 as the *Google News* embeddings. Context window size was set at 15, as the average length of the articles is 217 in the titled corpus and the sentences tend to be long. Therefore, we need to skip the stop words in the sentences by setting up a larger context window. In order to capture the infrequent words which can, later on, be the important features in categorization, we set the minimum frequency to one. The default sampling threshold 1e-5 was employed, as it is proven to be a rational frequency threshold to balance the rare and frequent words according to Mikolov et al. [2013b, 4]. Five "noise" samples (co-occurrence in the corpus) were drawn to compute the proxy of context closeness in the corpus. We did not use the paragraph vector (*DBOW*) in training because our goal was to compare the efficacy of pretrained and retrained word embeddings. 100 iterations of training with parallelization took about 40 minutes to finish on a server with 16 *Intel Xeon* processors and 500G of memory.

5.2 Partially Supervised Clustering of *Titled* Corpus

We have chosen to represent the document embeddings by averaging word embeddings, and we would test the vector composition proposed by Mitchell and Lapata [2008] during clustering. The pretrained and retrained word embeddings will be used as features in the k-means clustering. As stated in Section 4.3, the initialization of cluster centroids can influence the performance of clustering. We have also discussed how many clusters we should define for k-means clustering algorithm. According to the legal framework of IIAs, there are nine topics we can summarize with the concrete definitions from Salacuse [2015]. Recall IIAs are composed of various types of international agreements (e.g. BITs, TIPs), amongst which not only international investment was negotiated, but also other fields of cooperation (e.g. trade, technological) were included in IIAs. As a result, we should allow for another category "others" which cover all the articles that do not deal with issues about international investment. Consequently, the numbers of clusters we tested in experiments are 9, 10 and 11. We will explain later why we have tested three different numbers of clusters.

Table 12 gives an overview of possible settings we could test in the k-means clustering. With six categories of settings (the number of clusters, the type of word embeddings, weights for average, vector composition, initial centroids, passes), it

would take $3 \times 2 \times 2 \times 2 \times 2 \times 2 = 96$ runs to test all the possible combinations of settings. Obviously, brute-force approach to attempt every combination is not ideal for our experiments. As we look at the setting categories from left to right in Table 12, we can first test the number of clusters, the types of word embeddings and the weights of average. Without gold standards for the cluster labels, the evaluation of clustering is performed by observing the MDS visualization, calculating the Silhouette coefficient and interpreting the keywords from each cluster using LDA. After having selected the best number of clusters, the type of word embedding and the weighting scheme, we can continue with the other three settings.

no. cluster	word embedding	average	initial centroids	vector composition	passes
9	pretrained	no weights	random	addition	100
10	retrained	TF/IDF weights	topic definitions	weighted addition	200
11					

Table 12: Settings in partially supervised clustering

We first started with the number of clusters equal to ten. Figure 16(A), Figure 16(B), Figure 16(C), Figure 16(D) are the two-dimensional MDS plots for the clustering results with the randomly initialized centroids. The four figures show the clustering results with the features being the pretrained word embeddings (from *Google News*) with no weights in averaging, the retrained word embeddings with no weights in averaging, the pretrained word embeddings averaged by the TF/IDF weights and the retrained word embedding averaged by the TF/IDF weights. Please note the numbering and the coloring of the clusters were assigned by k-means randomly, which does not necessarily correspond to our final numbering of categories, nor does it remain consistent across different runs.

Weights in averaging: No weights or TF/IDF? We can interpret the figures by firstly looking at the partitions of points in space. In Figure 16(D), the red dotted cluster "dominates" the space of other clusters and is clearly inseparable with the other clusters. If we set the number of clusters to 11, with the assumption that there could be another "hidden topic" in our titled corpus, the TF/IDF-weighted word embedding generated some similar scenarios of point distribution in MDS, see Figure 17(A) and Figure 17(B). Figure 16(D) suffers less from this overlapping problem; however, the distribution of each cluster is also not clear-cut and distinguishable. To summarize, we can say that article embeddings computed by averaging word embeddings with TF/IDF weights have not worked well for our corpus. This is quite the opposite as proposed in the literature (see Lilleberg et al. [2015]). One

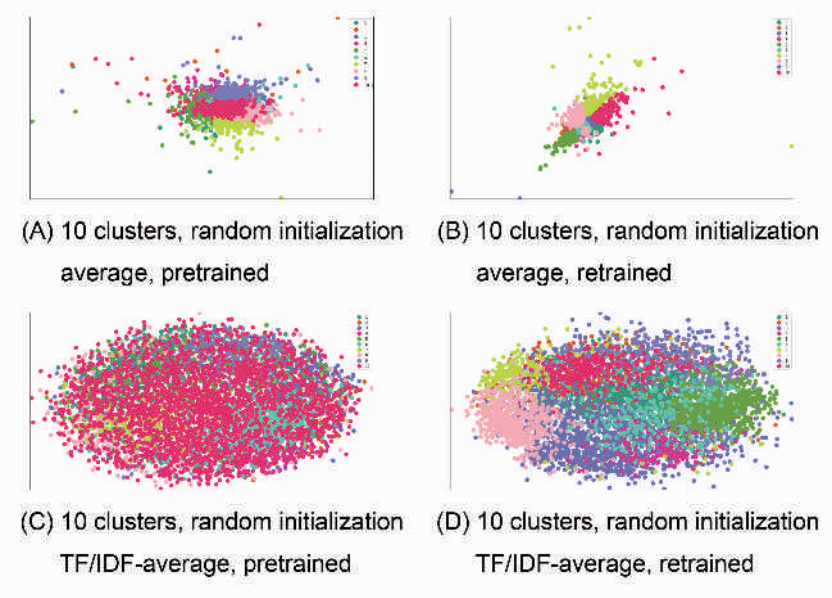


Figure 16: Comparison: four scenarios of ten clusters the with randomly initialized centroids

explanation why the method does not work in our experiment, is that the TF/IDF score computed from the titled corpus does not produce discriminative features in the clustering. Therefore, for the article clustering, we will take the averaged word embeddings in the articles as the reasonable representations of article embeddings.

Initialized or random centroids? Observing Figure 16(A) and Figure 16(B), although the points are distributed differently in each figure, the overlapping of classes are severe. This brings us to perform tests with the initialization of cluster centroids because the k-means algorithm is quite sensitive to the centroid initialization. So far we just looked at the MDS plots with random initialization and the overlapping of clusters is quite severe. Therefore, we made use of definitions summarized according to Salacuse [2015] for ten topic domains ("others" as the tenth category, the other nine can be found in Section 4.3.1, number 2-10). The definitions are also represented by averaging the embeddings of their compositional words. This means, we computed the vectors for the definitions in each topic and used those document vectors as the initialized centroids in clustering. The improvement of cluster partition is observable in Figure 18(A) and Figure 18(B). Figure 18(B) exemplifies a better scheme of cluster distribution compared to Figure 18(A) because the clusters are clearer in the former case and the clusters have divided the space into different areas. Compared with random initialization showed in Figure 16(A)(B), topic def-

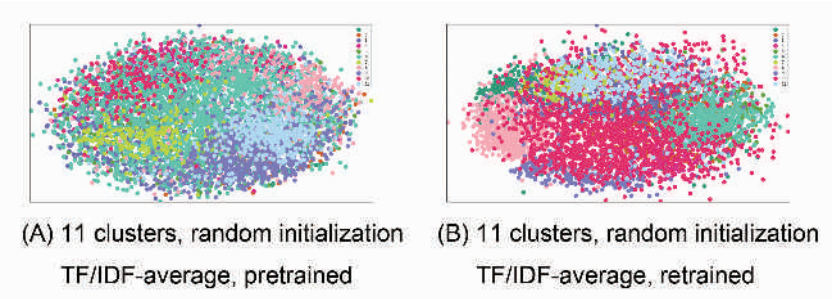


Figure 17: Comparison: two scenarios of eleven clusters with the randomly initialized centroids

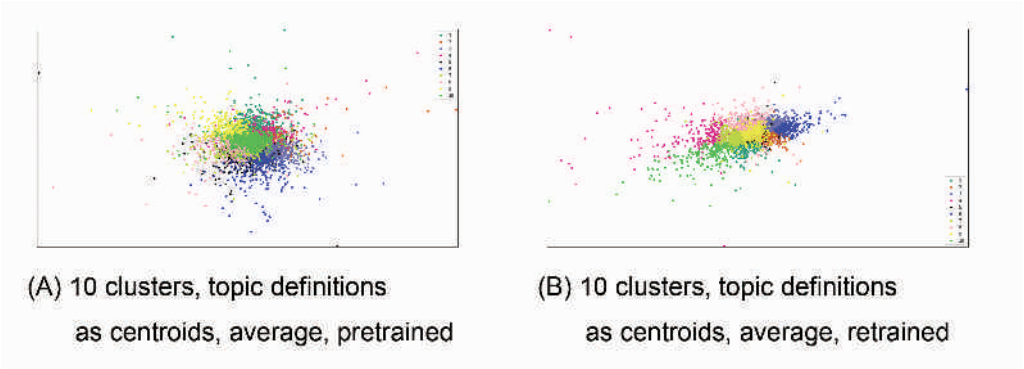


Figure 18: Comparison: two scenarios of ten clusters with the initialized centroids of definitions

initions as the initialized centroids lead to a tendency of clearer cluster partition. We can then conclude that k-means with initialized centroids of definitions in IIAs outperformed that with the randomly initialized centroids.

Keywords in clusters To further examine Figure 18(B), we applied LDA to each cluster of articles for 20 representative keywords of each topic. Then we can map the keywords with the given ten topic definitions. Interestingly, we cannot find two distinguishable sets of keywords for "expropriation and dispossession" and "losses from armed conflict or international disorder", and yet those two topics both involve certain issues of compensation (Salacuse [2015] put them as two topics, however!). One set of keywords we obtained described both of the topics under the same cluster, with terms such as "compensation", "expropriation", "protection", etc. It may first be due to the fact that there are not many articles about "losses from armed conflict or international disorder". Second, both topics deal with the compensation in international investment, regardless of the causes. Last but not least, we can see

from the keywords generated for other clusters, none of which could describe these two given topics properly. Moreover, within another set of keywords, two completely different topics were entangled, e.g. “dispute” and “termination” were mixed in the same set. To tackle this problem, we tested with the number of clusters equal to eleven, with the retrained word embeddings; nonetheless, the mapping of keywords between clusters and ten topics remained unsatisfactory.

Pretrained or retrained word embeddings? We compared the keywords generated by Figure 18(A)(B), the problems with the keyword mixture across clusters persisted. Moreover, the keywords produced with the pretrained word embeddings could identify fewer topics compared to those produced by the retrained embeddings. Hence, we decided to take the retrained word embeddings as our features in clustering because they were tailored to the SNIS corpus.

Until now, we noticed that we had to change the taxonomy of IIAs because in our corpus, one topic “losses from armed conflict or international disorder” is highly unevenly distributed, with only less than ten articles in the titled part. Hence, we decided to merge those two topics on compensation into one, i.e. “compensation (expropriation and dispossession/losses from armed conflict or internal disorder)”. Again we are back to the cluster number of nine.

Number of clusters As a control, we let the number of clusters be nine and run the retrained word embeddings setup with the initialized topic centroids. We can observe from the MDS visualization. In Figure 19, the partition of data points appeared different compared to Figure 18(B). Consequently, a careful examination of the keywords of nine clusters was needed. Except for two topic domains, the other seven topics matched the LDA-generated keywords nicely.

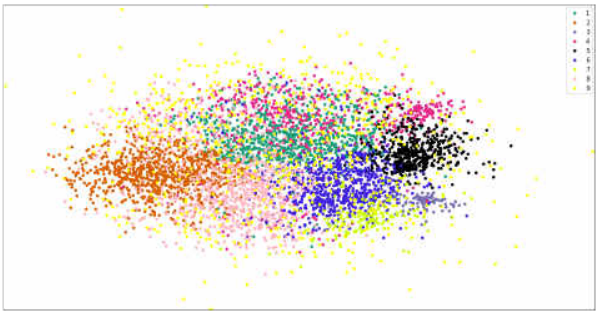


Figure 19: Nine clusters with topic definitions as the centroids represented by averaging the retrained word embeddings

One problematic topic contained keywords from“definitions and scope of applica-

tion" and "general standards of treatment of foreign investments and investors". We further investigated the article titles clustered in this topic. Indeed, many titles for articles on the scope of application were clustered in the topic. The remaining titles were mostly from the topics "general standards of treatment of foreign investments and investors" or "other (international investments irrelevant).

Another topic was summarized with keywords such as "committee", "procedure", "measure", "consultation", "application", "rule", "review", "joint", "standard", "transparency" and "objective". This set of words points to the international governance and regime in IIAs. Examination of the titles further proved that this topic was indeed about the organization matter in IIAs and involved many standards of international coordination and special committees. Therefore, we added another topic to our existing list of nine topics. So our final set of ten topics is shown in Table 13 below, whose definitions are listed in Appendix C.

cluster	topic
0	Conditions for the entry of foreign investment and investors
1	Definitions and scope of application
2	Others (other political, economical,cultural, technological and scientific cooperation)
3	Operational and other conditions
4	Treaty entry, exceptions, modifications and terminations
5	Dispute settlement
6	General standards of treatment of foreign investments and investors
7	Monetary transfers
8	Compensation (expropriation and dispossession/losses from armed conflict or internal disorder)
9	International governance and regime in IIAs

Table 13: Final list of topic in clustering

As the clustering results of nine clusters were not satisfactory (see Figure 19), with the number of clusters equal to ten, we ran the k-means clustering with the optimal settings we confirmed so far: averaging the retrained word embeddings with no weight, ten clusters, with topic definitions as the initialized centroids. We found that the mapping between the keywords and the list of ten topics as mentioned earlier was not completely mutually exclusive, i.e. some titles that should belong to the same cluster ended up in other clusters.

Vector composition of title and text In order to further improve the clustering results, we experimented with vector composition to represent the whole article. So far we just concatenated the title and text as a whole and then computed their document embeddings. We would like to test the vector composition strategies

cluster	count	keywords
0	761	“technical”, “regulation”, “transparency”, “assessment”, “procurement”, “conformity”
1	641	“general”, “application”, “scope”, “definition”
2	794	“cooperation”, “trade”, “economic”, “environment”, “social”, “technology”, “technical”, “industrial”, “drug”
3	606	“operation”, “material”, “person”, “intellectual”, “natural”, “business”, “temporary”, “access”, “border”, “cross”,“service”, “transport”
4	554	“force”, “entry”, “duration”, “termination”, “amendment”, “elimination”, “restriction”, “annexe”
5	537	“dispute”, “settlement”, “investor”, “investment”, “arbitration”, “state”, “interpretation”, “award”, “court”, “tribunal”, “claim”, “procedure”, “resolution”
6	667	“national”, “protection”, “promotion”, “nation”, “obligation”, “right”,
7	162	“transfer”, “capital”, “repatriation”, “payment”, “return”, “profit”, “income”, “guarantee”, “movement”, “current”, “free”, “revenue”, “asset”, “exchange”, “currency”, “fund”, “property”
8	127	“compensation”, “expropriation”, “loss”, “nationalization”, “damage”, “measure”, “territory”, “indemnification”, “war”, “property”, “nationalize”, “deprivation”, “dispossession”
9	446	“committee”, “consultation”, “joint”, “council”, “procedure”, “commission”, “implementation”, “rule”, “review”, “member”, “panel”, “meeting”, “cooperation”

Table 14: Keywords of ten clustered topics

for titles and texts. Addition and weighted addition strategies were chosen. For addition, we simply computed the $v_{article} = \alpha v_{title} + \beta v_{text}$ with $\alpha = \beta = 1$. For weighted addition, we tried two settings $\alpha = 0.5, \beta = 0.5$ and $\alpha = 0.2, \beta = 0.8$, to assign the weights of contributions of title and text. The article vector was computed by element-wise addition with the specified weights. It turned out that when $\alpha = \beta = 1$ we could identify the best mappings between keywords and the ten topics.

The best setting for k-means clustering The clusters of the best settings using k-means (ten clusters, the average retrained word embeddings, the additive vector composition, the initialized centroids with topic definitions) are visualized in Figure 20. We also tested the best settings with the iteration passes of 100 and 200 and found out that compared with those of 100 passes, the clustering results of 200 passes have not changed substantially regarding the Silhouette coefficient, the MDS cluster visualization, and the topic keywords. Hence, we report only the results of 100 passes. Table 14 lists the keywords in each cluster. The cluster numbers and their corresponding topics can be found in Table 13.

As we can see from Figure 20, the distribution of data points in the two-dimensional space exemplifies the characteristics of each topic and their relations with one an-

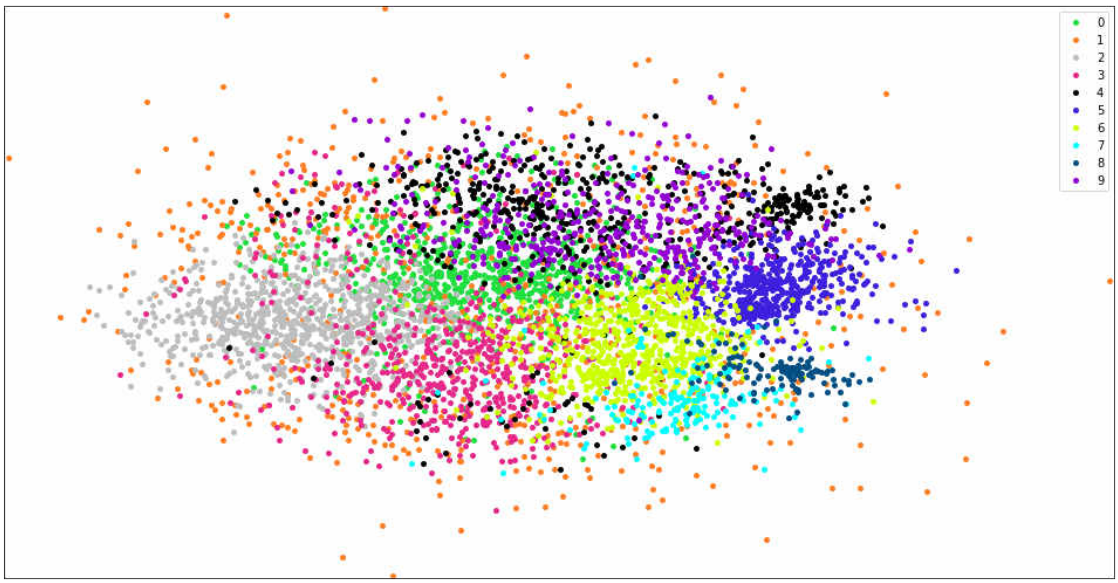


Figure 20: Best clustering settings for the titled articles: ten clusters, the average retrained word embeddings, the additive vector composition, the initialized centroids with topic definitions

other. We start from the left bottom, clusters 2, 0, 3, 6, 4, 8, 5 have very condensed intra-cluster distributions (from left to right: "others", "conditions for entry of foreign investment and investors", "operational and other conditions", "general standards of treatment", "monetary transfers", "compensation", "dispute settlement"). Data points belong to the seven clusters mentioned above group closer to their group members than to the data points from other topic clusters. The more fluid clusters are 9 and 4 (at the upper part of the figure, from left to right, "international governance and regime in IIAs", "treaty entry, exceptions, modifications, terminations"), which partially overlap. This can be explained by their textual and legal interconnectivity with each other. For instance, "international governance and regime" (cluster 9) covers the principles, norms, rules, decision-making processes of IIAs; the provisions of "entry, exceptions, terminations, modifications" (cluster 4) can intertwine with cluster 9 because both clusters deal with the institutional provisions. The remaining clusters cover mainly the negotiated terms and conditions on the concrete issues and matters of international investment and investors. There is one very scattered cluster 1 that spreads surrounding the other clusters. Cluster 1 indicates the topic "definitions and scope of application" where the definitions used throughout the treaties are explained, and the applicability of terms is specified. It is expected that definitions are composed of various terms which are then further specified in other articles of the same treaty. Therefore, cluster 1 can be rather

scattered in the semantic space.

word embedding	average	initial centroids	Silhouette coefficient
pretrained	no weights	random	0.037
pretrained	no weights	topic definitions	0.036
pretrained	TF/IDF weights	random	0.045
pretrained	TF/IDF weights	topic definitions	0.054
retrained	TF/IDF weights	random	0.100
retrained	TF/IDF weights	topic definitions	0.084
retrained	no weights	random	0.117
retrained	**no weights**	**topic definitions**	**0.104**

Table 15: The Silhouette coefficients of clustering with ten clusters, $v_{article} = v_{title} + v_{text}$

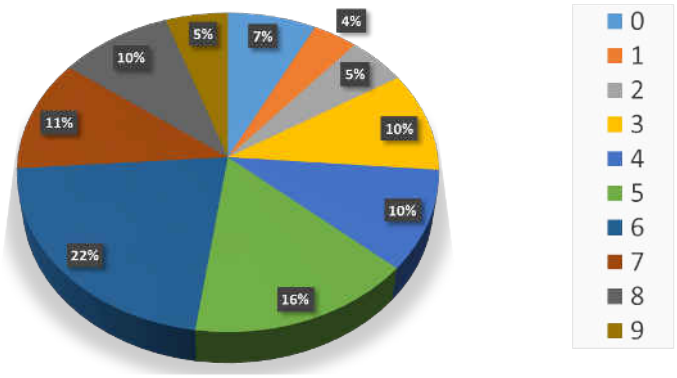


Figure 21: Pie chart for titled article% in each cluster

We can see from the percentage distribution of each cluster in Figure 21, with the total number of titles being 5,101. The numbering in the legend corresponds to the cluster numbers in Table 13. The cluster with the most articles is number 6 (22%), on "general standards of treatment of foreign investments and investors", followed by "dispute settlement" (number 5, 16%), "monetary transfers" (number 7, 11%), "treaty entry, exceptions, modifications and terminations" (number 4, 10%), "operational and other conditions" (number 3, 10%), "others (other political, economical, cultural, technological and scientific cooperation)" (number 8, 10%), "compensation" (number 0, 7%), "conditions for the entry of foreign investment and investors" (number 2, 5%), "international governance and regime in IIAs" (number 9, 5%) and "definitions and scope of application" (number 1, 4%).

Evaluation of the *titled* article clustering Another important benchmark to evaluate clustering is the Silhouette coefficient which measures the distance between the mean of the current cluster and the mean of its closest cluster. Theoretically speaking, the higher the score is, the better the clustering algorithm performs. Hence, we computed the Silhouette coefficients for the best settings of cluster number (ten) and the vector composition of title and text ($v_{article} = v_{title} + v_{text}$) (see Table 15). The best setting listed in Figure 20 has a Silhouette coefficient of 0.104 (bold). The other settings have mostly lower scores in the Silhouette coefficient. Only one setting has a slightly higher score. As we stated before, we cannot judge the clustering performance based on only one criterion. Three criteria (namely, the Silhouette coefficient, the distribution of clusters in MDS visualization and the mapping between the keywords and ten topics) must be examined at the same time. As a result, the best setting we chose is the only setting that suffices all three criteria.

To better understand the performance of k-means clustering in each cluster, we randomly selected 100 instances and obtained the annotations from the professional, so that we can compare the automatic output and the expert judgment. The overall accuracy of clustering is 50% across all clusters. From Table 16 we can see that accuracies in clusters 5, 7, 8 have reached 100%, followed by cluster 9 with an accuracy of 60%, cluster 6 with 50%, cluster 2 with 44.4% and cluster 4 with 33.33%. Only one instance out of cluster 1 fell into the correct cluster. Instances in clusters 0 and 3 have all been clustered wrongly. The clusters with the high accuracies have condensed clusters as shown in Figure 20.

cluster	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
gold	5	31	9	2	3	7	8	20	10	5
accurate	0	1	4	0	1	7	4	20	10	3
accuracy	0.00%	3.23%	44.44%	0.00%	33.33%	100.00%	50.00%	100.00%	100.00%	60.00%

Table 16: Accuracy for partially supervised clustering in each cluster for 100 titled instances

What are the false negatives in each cluster? How were they grouped by the clustering algorithm?

- False negatives in cluster 0 were all clustered as 6 ("general standards of treatments"). These articles were all titled with "promotion of investment". The expert also reported the difficulty in differentiating the two topics during his annotation.
- False negatives in cluster 1 were from almost every other cluster except clusters 7 and 8.

- In cluster 2, false negatives were mainly from cluster 3.
- False negatives in clusters 3 and 4 were mainly from cluster 0.
- There was no dominate cluster where the false negatives are from in rest of the clusters.
- Cluster 8 ("compensation") is the only cluster that does not produce false negatives for other clusters. From Figure 20 we can see that cluster 8 lies in the periphery of data points.

Note that we have a relatively large evaluation set for cluster 1, which does not correspond to the distribution of clusters in the titled articles as illustrated in Figure 21. For this reason, if we neglect cluster 1 in the evaluation, we would achieve an accuracy of $\frac{49}{100-31} = 71\%$. However, it is not particularly surprising that cluster 1 has a lower accuracy, given the fact that it has a fluid intra-cluster structure. Each instance within that cluster can be linked to other clusters by addressing the basic definitions and the scope of conditions for other pertaining articles. Therefore, it is crucial to define our goal in the evaluation of clustering: If we are only interested in certain clusters, we shall only observe the accuracy per cluster. To put it in another way, we should bear in mind that the accuracy of each cluster will directly influence the accuracy of text classification tasks.

5.3 Article Classification

With partially supervised clustering evaluated in Section 5.2, we managed to obtain the cluster labels from a set of ten IIA topics and map them to 34,524 title articles. With this pre-labeled data as the training and tuning sets (lemmatized and lower-cased, 34,524 articles), we ran six different classifiers on the data and hereby report the best settings of 5-fold cross-validated grid search of KNN, SVM, MLP, SGD and the best hyperparameter settings for CNN. The average training and testing accuracies of cross-validation were reported on the titled corpus.

1. `KNeighborsClassifier()`
   Best set of parameters and their values from cross-validation:
   `"n_neighbors":5, "weights":"distance", "algorithm":"ball_tree",`
   average training accuracy: 0.870, average test accuracy: 0.884.
2. `SVC()` and `LinearSVC()`
   Best set of parameters and their values for cross-validation for
   `SVC(kernel="linear"):"C":10, "gamma":0.001,` average training accuracy:

0.850, average test accuracy: 0.860;
Best set of parameters and their values for cross-validation for `LinearSVC (multi_class="ovr")`: `"penalty":"l2", "C":10.0, "class_weight": "balanced",` average training accuracy: 0.839.
Best set of parameters and their values for cross-validation for `SVC(kernel="rbf")`: `"gamma":2, "C":1`, average training accuracy: 0.883, average test accuracy: **0.893**.

The OvO SVM is better than the OvR under the same circumstance (here with the linear kernel).

3. `MLPClassifier()`
   Best set of parameters and their values from cross-validation: `"alpha":0.001, "activation":"relu", "hidden_layer_sizes":(100,),` average training accuracy: 0.856, average test accuracy: 0.874.

4. `SGDClassifier()`
   Best set of parameters and their values from cross-validation: `"penalty":l1, "alpha":1.0e-05, "loss":"hinge",` average training accuracy: 0.832, average test accuracy: 0.840.

5. `CNN`
   Best set of hyperparameters from 10-fold cross-validation: `embedding dimensions=50, filter size=(3,4,5), number of filters =20, dropout=0.1, batch size=64, number of epochs=10,` average test accuracy: 0.752[2].

Gaussian SVM has scored the highest (0.893) amongst the six classifiers in the evaluation of cross-validation. We can observe that for a classifier which does not use deep learning, the average test accuracy in the training set is slightly higher than the average training accuracy. This small increase can be explained by the textual similarity between the test and the training instances. The vocabulary in IIAs of the titled corpus was used quite frequently, with 176 times per type on average (see Table 6). The vocabulary is also large in the untitled part because the translation has introduced various alternative terms, with per type used only 40 times on average.

[2]For CNN, we also tested with the scripts written in `Keras` (https://keras.io/), with `TensorFlow` backend. The sample scripts can be found at https://github.com/alexander-rakhlin/CNN-for-Sentence-Classification-in-Keras (accessed 20 April 2017). However, there was a huge discrepancy in the accuracy computed by the `Keras` function `evalutate(x,y)` with the true labels y and the input x, and the accuracy computed using predictions output by the `Keras` function `predict(x)`. Hence, we decided not to take the results of `Keras`, although it has produced higher scores in training accuracy.

We tested the classifiers on our untitled part of the corpus (10,074 articles, lemmatized, lowercased). In order to evaluate the classifier performance on the untitled corpus, we randomly selected 100 untitled articles and obtained their human annotated labels from the expert annotation. Overall the results of supervised learning are not ideal, with the highest accuracy of 46% (by CNN) on the 100 evaluation, followed by Gaussian kernel SVM 15% (see Table 17). We analyzed the classification accuracy for each class in detail as shown in Table 18. The classes 5, 7, 8 have 100% accuracy which is in line with the training performance of partially supervised clustering in those classes (see Table 16). The other classes with high accuracy in text classification are classes 6 and 4.

With the randomly initialized word embeddings (50 dimensions) after ten epochs of training, we can already achieve a higher accuracy with a simple CNN classifier, in comparison with the traditional classifiers. Our findings seem to show that merely using the BoW features does not provide enough predicting power in classifying texts that belong to various topics from the same domain (IIAs in our case). Embeddings, the input matrix of CNN, expand the semantic space for the CNN classifier. Then the CNN classifier uses 20 filters (`size=(3,4,5)`) per document input matrix to compress textual information and generate more condensed representations of articles. Recall the average lengths of articles in our titled and untitled corpus are 217 and 160 tokens, respectively. When applying the filters and filter sizes, CNN scans through an average titled article with $20\times(\frac{217}{3}+\frac{217}{4}+\frac{217}{5}) = 20\times(73+55+44) = 3440$ features, through an average untitled article $20\times(\frac{160}{3}+\frac{160}{4}+\frac{160}{5}) = 20\times(54+40+32) = 2520$ features. This gives a better representation of the textual data because we incorporate contextual data up to 5-grams into our feature engineering.

Nonetheless, the CNN classifier has the tendency to predict class 6, as we can see that class 6 occupies a large part of the false negatives. It can be explained by the fact that class 6 might have the largest portion of testing materials in the test set. In the training set to generate class labels, cluster 6 is the largest class as demonstrated by Figure 21. The CNN classifier suffered severely from an imbalanced distribution of classes; thus, it has the tendency to assign the label from the largest class to the instances with which it was uncertain. Alternatively, we can improve our CNN classifier by adding balanced training materials in the training set, i.e. each class has the same number of instances for training.

On the contrary, the classifiers trained with `scikit-learn` [Pedregosa et al., 2011] suffered less severely from the imbalanced distribution of classes compared with CNN. It is because for the multiclass classification task, the default mode in the `SVC()` classifier (linear, Gaussian) is OvO because MLP works with a multinomial

distribution of each class which produces a probability distribution for all the given classes. SGD is the only classifier that uses the default OvR strategy; hence, it performed poorly with the imbalanced data comparing with the SVMs. It would be beneficial to apply the OvO meta-classifier (`multiclass.OneVsOneClassifier`)[3] to MLP and SGD classifier, so that we can make use of the majority vote of classifiers.

classifier	KNN	linear SVM	non-linear SVM	MLP	SGD	CNN
accuracy	4%	10%	15%	2%	11%	**46%**

Table 17: Overall results of accuracy for 100 untitled instances

class	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
gold	1	18	10	9	6	8	23	5	7	13
accurate	0	0	1	0	4	8	20	5	7	1
accuracy	0.00%	0.00%	10.00%	0.00%	66.67%	100.00%	86.96%	100.00%	100.00%	7.69%

Table 18: Accuracy of the CNN classifier per class for 100 untitled instances (46 accurate instances in total)

Based on the manual analysis of 100 untitled instances, because of the imbalance of class distribution in the 100 instances and the fact we are not aware of the true distribution of class in the untitled part of corpus, it is precarious to draw the conclusion that our experiments have not brought about insights on text classification of the ten topics. Conditioning the accuracy on certain classes where the representations of textual and legal information are less heterogeneous and more concise such as "dispute settlement" (5), "monetary transfer" (7) and "compensation" (8), the CNN classifier could deliver a large fraction of accurate predictions. However, the classifiers which do not utilize word embeddings fail to capture the textual representations of articles with the simple TF/IDF-transformed BoW features. As the BoW features can capture only the surface textual similarity and a minimal part of the distributional similarity through TF/IDF transformation in the text snippets, the classifiers using these features have weak prediction power in our learning problem. Despite using merely randomly initialized word embeddings, the simple CNN classifier compresses the context by moving the feature filters across the texts. This makes features of the CNN classifier rich of the distributional semantic representations of the scanned articles. In a nutshell, a simple CNN classifier has proven the efficacy of word embeddings, even if they have been randomly initialized and have the dimensionality of 50. Word embeddings have greatly expanded the semantic features of words and documents (articles) which can effectively reduce the sparsity

[3]http://scikit-learn.org/stable/modules/multiclass.html (accessed 10 May 2017).

in the feature representations using conventional context-counting methods [Baroni et al., 2014].

As a comparison with the text classification, we report the efficacy of using the retrained word embeddings from the titled corpus to cluster the untitled articles in the next section.

5.4 Partially Supervised Clustering of *Untitled* Corpus

We used the untitled part of the corpus (lowercased, lemmatized) in a setting of *partially supervised k-means clustering*. The goal is to cluster the untitled articles based on their document embeddings computed by averaging word embeddings retrained on the titled part of the corpus (described in Section 5.1), and to compare the accuracy in categorizing 100 untitled instances with that of the best supervised learning classifier (i.e. a CNN classifier).

The identical setting from Section 5.2 that achieved the best cluster partition was applied here again: ten clusters, the average retrained word embeddings, the additive vector composition, the initialized centroids with topic definitions. The resulting clusters are visualized in Figure 22.

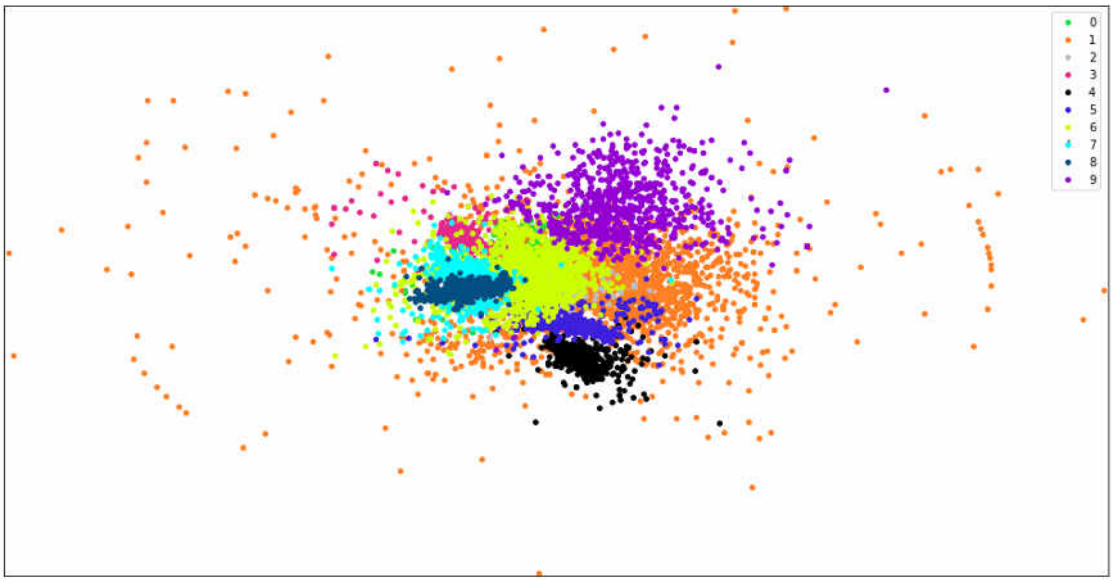


Figure 22: Best clustering settings for the untitled articles: ten clusters, the average retrained word embeddings, the additive vector composition, the initialized centroids with topic definitions

Similar to the results in the titled part of the corpus, articles from cluster 1 on "def-

initions and scope of application" spread across the space and encompass the other clusters. Clusters 0 ("entry of foreign investment and investors") and 2 ("others") are hidden behind cluster 6 ("general standards of treatment for foreign investment and investors"). The explanation for this might be that terms about foreign investment and investors can be covered both in cluster 0 or 6. For cluster 2, it can be due to the fact that there are not many instances from that cluster in the untitled part. The rest of the clusters have condensed intra-cluster distributions and are allocated nicely in the space. Another perspective to interpret the results is to look at the language quality and consistency. As shown in Table 7, category 3 (for good translations) has a higher portion in the untitled part (23%) than that in the titled part (17%). Translation renders more condensed representations, as the SMT systems translate from foreign languages into English consistently.

cluster	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
gold	1	18	10	9	6	8	23	5	7	13
accurate	0	7	7	3	5	7	10	3	7	11
accuracy	0.00%	38.89%	70.00%	33.33%	83.33%	87.50%	43.48%	60.00%	100.00%	84.62%

Table 19: Accuracy of the k-means clustering per cluster for 100 untitled instances (60 accurate instances in total)

5.5 Comparison: Article Classification vs. Article Clustering

We evaluated the clustering of the untitled articles with 100 annotated untitled instances. The overall accuracy of prediction has reached 60%, with an increase of 30.4% compared with 46% of accuracy achieved by the CNN learner ($\frac{60-46}{46} = 30.4\%$). As we can see from the accuracy of each cluster in Table 19, the accuracies across clusters have increased systematically compared with Table 18, at the cost of certain high accurate clusters such as 5 and 7. It is encouraging to observe that the k-means clustering with the retrained word embeddings can improve the prediction in the fluid clusters such as 1 and 9. It also outperforms CNN in clusters with which the latter had difficulties, i.e. cluster 2.

Moreover, we also computed the keywords for each cluster which show the similar mapping patterns between the keywords and the ten topics as our mappings summarized in Table 14. Finally, the k-means clustering does not have a tendency of predicting a particular cluster label because it predicts the membership for clusters based on the intra-cluster similarity and inter-cluster dissimilarity.

The different features we used in CNN and k-means have reflected partially the distinction we made in Section 2.3 about lexical similarity, distributional similarity and word embedding similarity. For a better understanding of the difference of surface lexical similarity and word embedding similarity, we computed the average of the Jaccard distance and the normalized Levenshtein distance (see Section 2.3.1) for each cluster, in the evaluation sets with 100 annotated instances for the titled and untitled parts, respectively. The x-axis in Figure 23 denotes the cluster label 0-9. The y-axis in the average lexical distance of texts (stop words filtered, lemmatized and lowercased) measured by the Levenshtein and Jaccard measures.

It is obvious that even the articles belong to the same cluster have high lexical dissimilarity (higher than 70%). The cluster that contains the most similar texts is cluster 1 ("definitions and scope of application"). Regarding surface textual similarity, articles in cluster 1 share almost the same syntactic structures in introducing the concepts in IIAs, such as "for the purpose of this agreement : (1) the term "investment" mean, ...; the term "investor" means: ...". However, any important term in IIAs can be defined with this structure, e.g. for intellectual property rights. The meaning of the texts varies largely from one definition to another. Therefore, to categorize articles from cluster 1, it requires not only the semantic expansions, as well as certain syntactic input. It is well known that word embeddings can capture certain linear semantic and syntactic regularities [Mikolov et al., 2013c]; as a result, the k-means clustering which makes use of document embeddings has the advantage over the context-counting classifiers that learn from the BoW model. Because of this, the k-means algorithm outperformed the simple CNN in cluster 1 with an increase of accuracy by 39%.

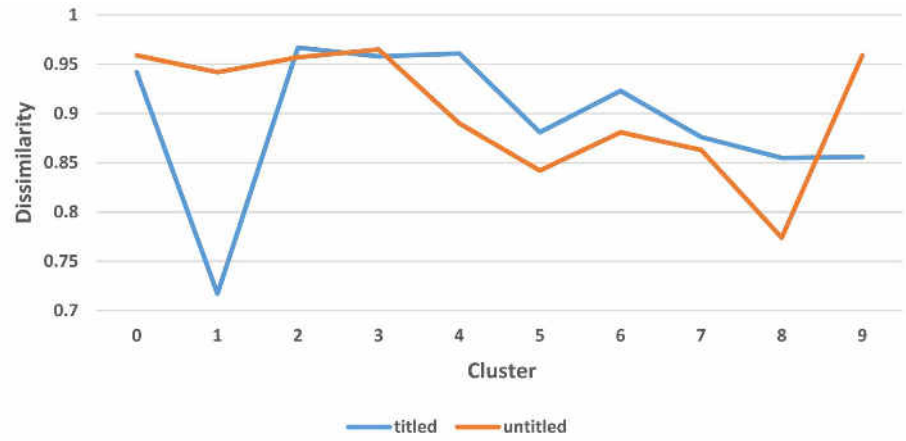


Figure 23: Average lexical dissimilarity of the titled and untitled evaluation sets

In Figure 23, we also find out that for certain clusters (5, 7, 8) where the technical

jargons are of particular use to the topics (e.g. jargons on "dispute settlement", on "monetary transfer") and rarely intertwine with other topics, both the semi-supervised learner (k-means with embeddings) and the supervised learner (CNN) can reach high accuracy. Furthermore, the supervised method has been even slightly better than the k-means clustering, because the lexical items in those clusters do not vary from article to article largely. Last but not least, we confirm that retrained word embeddings customized to our SNIS corpus are better than randomly initialized embeddings, especially when our corpus is relatively small.

To summarize the features we have used for various estimators in the supervised and semi-supervised settings, we map the textual similarity measures to the estimators and evaluate the efficacy of feature extraction in IIA text categorization in general. Table 20 shows the various feature engineering techniques that focus on different aspects of textual similarity (lexical, distributional, embedding) and their applicability in the estimators we used in our experiments. Likewise, this summary reflects our discussion above in that the estimators (e.g. CNN and k-means) that utilize word embeddings have the strongest predicting power as opposed to those classifiers which use only surface lexical and some distributional features.

		lexical	**distributional**	**embedding**
supervised	KNN	x	x	
	SVM	x	x	
	SGD	x	x	
	MLP	x	x	
	CNN		x	x
semi-supervised	k-means		x	x

Table 20: Summary of the interplay of textual similarity and text categorization, "x" = feature engineering techniques

6 Conclusion

This thesis is an endeavor devoted to an interdisciplinary research topic: how to better understand the inherent structures of IIAs. As the first step to explore the structure of IIAs with ten topics, this work has enhanced our understanding of the applicability of text categorization, be it classification or clustering, to capture the inherent content structure.

We put together an extensive literature overview on textual similarity (surface, distributional, embedding) and its applicability to text categorization. Whereas previous studies on IIAs have mainly focused on the level of the treaty, considerable progress has been made in this thesis about extending the unit of analysis to treaty articles. We have devised a pipeline which extracted and preprocessed the titled and untitled articles (34,524 and 10,047 snippets, respectively) from 2,823 treaties in the SNIS corpus. In order to expand the word semantics in our domain-specific corpus, we retrained the word embeddings with the pretrained embeddings from the *Google News* corpus. We then performed partially supervised clustering where we compressed the document semantics of 5,101 unique formal titles and their corresponding texts and then generated the article labels (out of the ten topics) for the titled part of the corpus.

We then trained six supervised classifiers on the titled corpus (labeled by ten topics) and tested them with the untitled corpus in a multiclass setting (ten classes). As a comparison with supervised learning, we tested the clustering methods on assigning labels for the untitled articles with the retrained word embeddings as features. Having access to an annotated evaluation set of 100 untitled articles, we compared the efficacy of the supervised and semi-supervised techniques in the same learning problem. We discussed the overall performance regarding accuracy for all estimators. Additionally, we compared the estimators' performance in each topic. This led us to the conclusion that the k-means clustering with the retrained word embeddings customized to the SNIS corpus has brought about an increase of 30% in accuracy compared to a simple CNN classifier which has outperformed the other five supervised learners.

This thesis has highlighted the importance of expanding the semantic features of documents in text categorization. We have obtained better results by converting documents into vector representations and utilizing the retrained word embeddings. Taken together, our findings suggest the important roles of the word and document embeddings in text categorization. The present findings have important implications for improving the supervised classifier: We could use the retrained word embeddings as features in a supervised setting; we could initialize the word embeddings in a CNN classifier with our retrained representations.

We hope that this work will be beneficial to the construction of IIA database in the future because it has tested different techniques to decipher the structure of IIAs by categorizing text snippets into the interlinking topics from the same domain.

7 Future Work

Due to time constraint, we did not test extensively whether the document embeddings trained with `doc2vec` which utilize the retrained word embeddings of the SNIS corpus can generate better representations of articles in the semantic space. Further studies, which use other vector composition strategies proposed by Mitchell and Lapata [2008] will need to be undertaken. It should also be examined, whether we can perform partially supervised clustering with the whole corpus (the titled and untitled parts included) and assign the labels for the untitled articles based on their cluster membership, as we have access to the article titles of the titled part. It would also be interesting to compare the output of affinity propagation (AP) (where no number of clusters should be specified) and that of the k-means clustering.

We have also found out that the CNN classifier and the k-means clustering perform differently in various topics. It remains to be tested if topic-specific learning techniques should be devised to tackle the variability of semantics and syntax in each topic. It would be worth testing whether using the annotated texts (e.g. PoS tagged, syntactically parsed) can improve the text categorization. These topics are reserved for our future work.

Last but not least, a hierarchical topic taxonomy has been proposed by UNCTAD[1]. The design and development of a system using hierarchical classification (see Silla Jr and Freitas [2011]) or clustering (see Sarkar [2016, 297]) is of interest to both communities of CL and IIAs.

[1] http://investmentpolicyhub.unctad.org/IIA/mappedContent (accessed 20 May 2017).

Glossary

accuracy The percentage of the accurately predicted labels according to the gold standards.

content words Words that have meaning, such as nouns, verbs, adjectives, adverbs.

corpus A collection of texts.

deep learning A machine learning technique with artificial neural network of more than one hidden layers.

gold standard The true labels generated by human annotations with a high inner-annotator agreement.

hyperparameter Different from *parameter*, hyperparameters cannot be directly learn from the training, such as the number of clusters in a k-means clustering. We usually *tune* the hyperparameters (*hyperparameter tuning, parameter tuning*).

lemma The canonical form of a word. For instance, word forms "eat", "ate", "eating", "eaten" share the same lemma "eat". Finding the lemma given a word form is called *lemmatization*. A tool which performs lemmatization is a *lemmatizer*.

loss function A function computes the difference between the predicted labels and the true labels, also called *cost function, objective function*.

machine learning A technique to learn from existing data and to predict.

machine translation A technique to translate text or speech from one language to another. *Statistical machine translation (SMT)* generates the translations based on statistical methods and bilingual corpora.

neural network *Artificial neural network* inspired by biology utilizes the connectivity of *neurons* to perform machine learning.

parameter A set of model parameters learnable from a machine learning setting, such as mean, standard deviation.

Part-of-Speech (PoS) tagging A process of finding the word type (e.g. verb, noun) given a word form.

penalty Penalty regulates the power of prediction in machine learning, hence also called *regularization*.

precision The fraction of relevant instances among the retrieved instances[2].

recall The fraction of relevant instances that have been retrieved over total relevant instances[3].

stop words Words that have little lexical meaning, as opposed to *content words*. In linguistics, they are also called *function words*.

[2]https://en.wikipedia.org/wiki/Precision_and_recall (accessed 10 June 2017).

[3]https://en.wikipedia.org/wiki/Precision_and_recall (accessed 10 June 2017).

References

C. C. Aggarwal and C. Zhai. *Mining Text Data*. Springer Science & Business Media, 2012.

C. C. Aggarwal, S. C. Gates, and P. S. Yu. On using partial supervision for text categorization. *IEEE Transactions on Knowledge and Data Engineering*, 16(2): 245–255, 2004.

E. Agirre, C. Banea, D. Cer, M. Diab, A. Gonzalez-Agirre, R. Mihalcea, G. Rigau, and J. Wiebe. Semeval-2016 task 1: Semantic textual similarity, monolingual and cross-lingual evaluation. *Proceedings of SemEval*, pages 497–511, 2016.

W. Alschner and D. Skougarevskiy. Treaty texts as data-developing new tools for negotiators and litigators to compare bilateral investment treaties. In *Legal Knowledge and Information Systems - JURIX 2015: The 28th Annual Conference*, pages 141–144, 2015.

W. Alschner and D. Skougarevskiy. Mapping the universe of international investment agreements. *Journal of International Economic Law*, pages 561–588, 2016a.

W. Alschner and D. Skougarevskiy. Rule-takers or rule-makers? a new look at african bilateral investment treaty practice. Technical Report 7, World Trade Institute (University of Berne), Swiss National Centre of Competence in Research, 6 2016b.

M. Baroni, G. Dinu, and G. Kruszewski. Don't count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors. In *Proceedings of the 52nd Annual Meeting of the ACL*, pages 238–247, 2014.

R. Bartolini, A. Lenci, S. Montemagni, V. Pirrelli, and C. Soria. *Automatic Classification and Analysis of Provisions in Italian Legal Texts: A Case Study*, pages 593–604. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004.

S. Bird, E. Klein, and E. Loper. *Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit*. O'Reilly Media, Inc., 2009.

D. M. Blei. Probabilistic topic models. *Communications of the ACM*, 55(4):77–84, Apr. 2012.

D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

E. de Maat and R. Winkels. Automatic classification of sentences in Dutch laws. In E. F. et al., editor, *Legal Knowledge and Information Systems*, volume 189, pages 207–216. IOS Press, 2008.

E. de Maat and R. Winkels. A next step towards automated modelling of sources of law. In *Proceedings of the 12th International Conference on Artificial Intelligence and Law*, pages 31–39, New York, NY, USA, 2009. ACM.

E. de Maat and R. Winkels. *Automated Classification of Norms in Sources of Law*, pages 170–191. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

R. W. Fasold and J. Connor-Linton. *An Introduction to Language and Linguistics*. Cambridge University Press, 2014.

J. R. Firth. A synopsis of linguistic theory, 1930-1955. *Studies in Linguistic Analysis*, 1957.

G. H. Golub and C. F. Van Loan. *Matrix Computations*, volume 3. JHU Press, 2012.

T. Gonçalves and P. Quaresma. Is linguistic information relevant for the classification of legal texts? In *Proceedings of the 10th International Conference on Artificial Intelligence and Law*, pages 168–176, New York, NY, USA, 2005. ACM.

J. Grimmer and B. M. Stewart. Text as data: The promise and pitfalls of automatic content analysis methods for political texts. *Political Analysis*, 2013.

D. Jurafsky and J. H. Martin. *Speech and Language Processing*. Prentice Hall, 2nd edition, 2009.

Y. Kim. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, page 1746–1751, Doha, Qata, 2014. ACL.

P. Koehn. Europarl: A parallel corpus for statistical machine translation. In *Proceedings of the MT Summit 2005*, pages 79–86, 2005.

P. Koehn. *Statistical Machine Translation*. Cambridge University Press, 2009.

J. H. Lau and T. Baldwin. An empirical evaluation of doc2vec with practical insights into document embedding generation. In *Proceedings of the 1st Workshop on Representation Learning for NLP*, pages 78–86, Berlin, Germany, August 2016. ACL.

Q. V. Le and T. Mikolov. Distributed representations of sentences and documents. In *Proceedings of the 26th International Conference on Machine Learning (ICML)*, volume 14, pages 1188–1196, 2014.

D. D. Lee and H. S. Seung. Algorithms for non-negative matrix factorization. In *Advances in Neural Information Processing Systems*, pages 556–562, 2001.

J. Lilleberg, Y. Zhu, and Y. Zhang. Support vector machines and word2vec for text classification with semantic features. In *IEEE 14th International Conference on Cognitive Informatics & Cognitive Computing (ICCI*CC)*, pages 136–140. IEEE, 2015.

E. Loper and S. Bird. NLTK: The natural language toolkit. In *Proceedings of the 2nd ACL Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics*, volume 1, pages 63–70, Stroudsburg, PA, USA, 2002. ACL.

C. D. Manning and H. Schütze. *Foundations of Statistical Natural Language Processing*. The MIT Press, Cambridge, Massachusetts; London, England, 2000.

D. Merkl and E. Schweighofer. En route to data mining in legal text corpora: Clustering, neural computation, and international treaties. In *Proceedings of the 8th International Workshop on Database and Expert Systems Applications*, pages 465–470. IEEE, 1997.

T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. 2013a. URL `https://arxiv.org/abs/1301.3781`.

T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119, 2013b.

T. Mikolov, W.-t. Yih, and G. Zweig. Linguistic regularities in continuous space word representations. In *Proceedings of 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (HLT-NAACL)*, volume 13, pages 746–751, 2013c.

J. Mitchell and M. Lapata. Vector-based models of semantic composition. In *Proceedings of the 8th ACL: Human Language Technology Conference (HLT)*, pages 236–244, Columbus, Ohio, USA, June 2008.

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

L. Pinto and A. Melgar. A classification model for portuguese documents in the juridical domain. In *The 11th Iberian Conference on Information Systems and Technologies (CISTI)*, pages 1–4, June 2016.

S. Raschka. *Naive Bayes and Text Classification I - Introduction and Theory*, 2014. URL `https://arxiv.org/pdf/1410.5329.pdf`.

S. Raschka. *Python Machine Learning*. Packt Publishing, Birmingham, UK, 2015. ISBN 1783555130.

J. D. Rennie, L. Shih, J. Teevan, D. R. Karger, et al. Tackling the poor assumptions of naive bayes text classifiers. In *Proceedings of the 20th International Conference on Machine Learning (ICML)*, pages 616–623, Washington DC, 2003.

X. Rong. *word2vec Parameter Learning Explained*. University of Michigan, 2014. URL `https://arxiv.org/abs/1411.2738`.

J. W. Salacuse. *The Law of Investment Treaties*. OUP Oxford, 2015.

C. Sammut and G. I. Webb. *Encyclopedia of Machine Learning*. Springer Science & Business Media, 2011.

D. Sarkar. *Text Analytics with Python: A Practical Real-world Approach to Gaining Actionable Insights from Your Data*. Apress, 2016.

H. Schmid. Probabilistic part-of-speech tagging using decision trees. In *Proceedings of International Conference on New Methods in Language Processing, Manchester, UK*, 1994.

E. Schweighofer, A. Rauber, and M. Dittenbach. Automatic text representation, classification and labeling in european law. In *Proceedings of the 8th International Conference on Artificial Intelligence and Law*, pages 78–87, New York, NY, USA, 2001. ACM.

C. N. Silla Jr and A. A. Freitas. A survey of hierarchical classification across different application domains. *Data Mining and Knowledge Discovery*, 22(1-2): 31–72, 2011.

D. Soutner and L. Müller. Continuous distributed representations of words as input of LSTM network language model. In *International Conference on Text, Speech, and Dialogue (TSD)*, 2014.

K. Sugisaki, M. Volk, R. Polanco, W. Alschner, and D. Skougarevskiy. Building a corpus of multi-lingual and multi-format international investment agreements. In *Legal Knowledge and Information Systems - JURIX 2016: The 29th Annual Conference*, pages 203–206, 2016.

M. A. Sultan, S. Bethard, and T. Sumner. Dls@cu: Sentence similarity from word alignment and semantic vector composition. In *Proceedings of the 9th International Workshop on Semantic Evaluation*, pages 148–153, 2015.

Y. Zhang and B. Wallace. A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. 2015. URL `https://arxiv.org/abs/1510.03820`.

Curriculum Vitae

Personal details

Family name	Rao
English name	Susie
First name	Xi
Address	Saegeweg 6, 8166 Niederweningen, Switzerland
Date of birth	May 22, 1986
Email address	xi.rao@uzh.ch

Education

2014 – 2017	Master of Arts on Multilingual Text Analysis and Computational Linguistics, Institute of Computational Linguistics, University of Zurich
2013 – 2014	German Goethe C1 Level Education, School of Applied Linguistics, Zurich University of Applied Sciences ZHAW
2009 – 2013	Political Economy and Conflict Risk Management, Center for Comparative and International Studies, ETH Zurich
2005 – 2009	Bachelor of Laws on International Relations and Economics, School of International Studies, Renmin University of China

Relevant Professional Activities

2016 – present	Research Assistant, Chair of Applied Economics, KOF Swiss Economic Institute, ETH Zurich
2013 – 2015	Project Assistant, School of Business, University of Applied Sciences and Arts Northwestern Switzerland FHNW
2009 – 2011	Research Assistant, Crisis and Risk Network, ETH Zurich

Publications and Presentations

Jancso, Anna; Rao, Xi; Graën, Johannes; Ebling, Sarah (2016).
A Web Application for Geolocalized Signs in Synthesized Swiss German Sign Language.
In: Proceedings of the International Conference of Computers Helping People with Special Needs (ICCHP), Linz, Austria, 13 - 15 July 2016.

Rao, Xi; Parijat Ghoshal (2016).
Normalization of Shorthand Forms in French Text Messages Using Word Embedding and Machine Translation.
Presentation at the International Symposium Parallel Corpora: Creation and Applications (PaCor), Santiago de Compostela, Spain, 1 - 3 Dec 2016 (publication forthcoming 2018).

A Tables

We list the three-letter codes as defined in ISO 3166-1[1], their corresponding contracting parties and the counts of negotiated treaties in the SNIS English corpus. The tables are sorted alphabetically by the three-letter codes from A to Z.

	code	contracting party	frequency		code	contracting party	frequency
1	ACP	African, Caribbean, and Pacific Group of States	1	107	KHM	Cambodia	23
2	AFG	Afghanistan	3	108	KOR	Republic of Korea	102
3	AGO	Angola	5	109	KWT	Kuwait	55
4	ALB	Albania	41	110	LAIA	Latin American Integration Association	1
5	ANCOM	Andean Community	3	111	LAO	Lao People's Democratic Republic	24
6	ARE	United Arab Emirates	38	112	LAS	League of Arab States	3
7	ARG	Argentina	57	113	LBN	Lebanon	50
8	ARM	Armenia	38	114	LBR	Liberia	4
9	ASEAN	Association of Southeast Asian Nations	10	115	LBY	Libya	23
10	ATG	Antigua and Barbuda	2	116	LCA	Saint Lucia	2
11	AU	African Union	1	117	LIE	Liechtenstein	1
12	AUS	Australia	35	118	LKA	Sri Lanka	28
13	AUT	Austria	50	119	LSO	Lesotho	3
14	AZE	Azerbaijan	37	120	LTU	Lithuania	49
15	BDI	Burundi	5	121	LVA	Latvia	40
16	BEL	Belgium	1	122	MAC	Macao Special Administrative Region, China	3
17	BEN	Benin	13	123	MAR	Morocco	62
18	BFA	Burkina Faso	11	124	MDA	Republic of Moldova	39
19	BGD	Bangladesh	32	125	MDG	Madagascar	7
20	BGR	Bulgaria	60	126	MDV	Maldives	1
21	BHR	Bahrain	25	127	MERCOSUR	Southern Common Market	3
22	BIH	Bosnia and Herzegovina	39	128	MEX	Mexico	36
23	BIMSTEC	Bay of Bengal Initiative for Multi-Sectoral Technical and Economic Cooperation	1	129	MHL	Marshall Islands	1
24	BLEU	Belgium–Luxembourg Economic Union	95	130	MKD	The former Yugoslav Republic of Macedonia	28
25	BLR	Belarus	57	131	MLI	Mali	6
26	BLZ	Belize	5	132	MLT	Malta	24

Table 21: The three-letter country codes, the contracting parties, and frequencies (part 1)

[1] See ISO 3166-1 standard at https://en.wikipedia.org/wiki/ISO_3166-1_alpha-3 (accessed 10 Jan 2017). The full names of contracting parties were retrieved via the information from *Standard country or area codes for statistical use (M49): Overview* at https://unstats.un.org/unsd/methodology/m49/overview/ (accessed 05 Feb 2017) and *List of intergovernmental organizations participating in the activities of UNCTAD* at http://unctad.org/meetings/en/SessionalDocuments/tdigolistd10_en.pdf (accessed 05 Feb 2017).

	code	contracting party	frequency		code	contracting party	frequency
27	BOL	Bolivia (Plurinational State of)	24	133	MMR	Myanmar	7
28	BRA	Brazil	10	134	MNE	Montenegro	15
29	BRB	Barbados	9	135	MNG	Mongolia	38
30	BRN	Brunei Darussalam	7	136	MOZ	Mozambique	17
31	BWA	Botswana	7	137	MRT	Mauritania	9
32	CACM	Central American Common Market	3	138	MUS	Mauritius	38
33	CAF	Central African Republic	3	139	MWI	Malawi	4
34	CAN	Canada	51	140	MYS	Malaysia	57
35	CARICOM	Caribbean Community, Regional Integration	7	141	NAM	Namibia	7
36	CEFTA	Central European Free Trade Agreement	1	142	NER	Niger	2
37	CEPGL	Central European Free Trade Agreement	1	143	NGA	Nigeria	22
38	CHE	Switzerland	122	144	NIC	Nicaragua	19
39	CHL	Chile	62	145	NLD	Netherlands	102
40	CHN	China	140	146	NOR	Norway	15
41	CIV	Cote d'Ivoire	8	147	NPL	Nepal	6
42	CMR	Cameroon	16	148	NZL	New Zealand	13
43	COD	Democratic Republic of the Congo	8	149	OCT	Overseas Countries and Territories	1
44	COG	Congo	7	150	OIC	Organisation of Islamic Cooperation	1
45	COL	Colombia	19	151	OMN	Oman	26
46	COM	Comoros	5	152	PAK	Pakistan	51
47	COMESA	Common Market for Eastern and Southern Africa	3	153	PAN	Panama	25
48	CPV	Cabo Verde	3	154	PER	Peru	38
49	CRI	Costa Rica	23	155	PHL	Philippines	38
50	CUB	Cuba	37	156	PNG	Papua New Guinea	6
51	CYP	Cyprus	21	157	POL	Poland	52
52	CZE	Czechia	86	158	PRK	Democratic People's Republic of Korea	13
53	DEU	Germany	88	159	PRT	Portugal	39
54	DJI	Djibouti	4	160	PRY	Paraguay	19
55	DMA	Dominica	2	161	PSE	State of Palestine	5
56	DNK	Denmark	55	162	QAT	Qatar	28
57	DOM	Dominican Republic	13	163	ROU	Romania	81
58	DZA	Algeria	24	164	RUS	Russian Federation	72
59	EAC	East African Community	2	165	RWA	Rwanda	6
60	ECCAS	Economic Community of Central African States	1	166	SACU	Southern African Customs Union	2
61	ECO	Economic Cooperation Organization	1	167	SADC	Southern African Development Community	3
62	ECOWAS	Economic Community of West African States	4	168	SAFTA	South Asian Free Trade Area	1
63	ECT	Energy Charter Treaty	1	169	SAU	Saudi Arabia	16
64	ECU	Ecuador	24	170	SDN	Sudan	20
65	EEU	Eurasian Economic Union	2	171	SEN	Senegal	18
66	EFTA	European Free Trade Association	26	172	SGP	Singapore	46
67	EGY	Egypt	96	173	SLE	Sierra Leone	2
68	ERI	Eritrea	3	174	SLV	El Salvador	20
69	ESP	Spain	76	175	SMR	San Marino	5
70	EST	Estonia	25	176	SOM	Somalia	1
71	ETH	Ethiopia	27	177	SPARTECA	South Pacific Regional Trade and Economic Co-operation Agreement	1
72	EU	European Union	64	178	SRB	Serbia	46

Table 22: The three-letter country codes, the contracting parties, and frequencies (part 2)

	code	contracting party	frequency		code	contracting party	frequency
73	FIN	Finland	59	179	SUR	Suriname	2
74	FJI	Fiji	1	180	SVK	Slovakia	51
75	FRA	France	99	181	SVN	Slovenia	36
76	GAB	Gabon	10	182	SWE	Sweden	70
77	GBR	United Kingdom of Great Britain and Northern Ireland	104	183	SWZ	Swaziland	4
78	GCC	Gulf Cooperation Council	6	184	SYC	Seychelles	1
79	GEO	Georgia	30	185	SYR	Syrian Arab Republic	30
80	GHA	Ghana	20	186	TCD	Chad	9
81	GIN	Guinea	15	187	TGO	Togo	3
82	GMB	Gambia	11	188	THA	Thailand	44
83	GNQ	Equatorial Guinea	5	189	TJK	Tajikistan	31
84	GRC	Greece	41	190	TKM	Turkmenistan	20
85	GRD	Grenada	2	191	TLS	Timor-Leste	1
86	GTM	Guatemala	18	192	TON	Tonga	1
87	GUY	Guyana	4	193	TTO	Trinidad and Tobago	12
88	HKG	Hong Kong Special Administrative Region, China	19	194	TUN	Tunisia	34
89	HND	Honduras	10	195	TUR	Turkey	100
90	HRV	Croatia	56	196	TWN	Taiwan, China	21
91	HTI	Haiti	4	197	TZA	United Republic of Tanzania	15
92	HUN	Hungary	56	198	UGA	Uganda	14
93	IDN	Indonesia	67	199	UKR	Ukraine	63
94	IND	India	91	200	UMA	Arab Maghreb Union	1
95	IRL	Ireland	1	201	URY	Uruguay	28
96	IRN	Iran (Islamic Republic of)	31	202	USA	United States of America	105
97	IRQ	Iraq	5	203	UZB	Uzbekistan	47
98	ISL	Iceland	11	204	VCT	Saint Vincent and the Grenadines	2
99	ISR	Israel	41	205	VEN	Venezuela (Bolivarian Republic of)	31
100	ITA	Italy	60	206	VNM	Viet Nam	52
101	JAM	Jamaica	12	207	VUT	Vanuatu	1
102	JOR	Jordan	48	208	WAEMU	West African Economic and Monetary Union	1
103	JPN	Japan	41	209	YEM	Yemen	27
104	KAZ	Kazakhstan	43	210	ZAF	South Africa	46
105	KEN	Kenya	6	211	ZMB	Zambia	7
106	KGZ	Kyrgyzstan	29	212	ZWE	Zimbabwe	18

Table 23: The three-letter country codes, the contracting parties, and frequencies (part 3)

B Sample Data, Scripts, and Annotations

B.1 Sample XML Documents of IIAs in Four Categories

(D1) Category 1 (*EN_HTML_STRUCTURED*): {ALB,CHN}_1993-02-13.xml, {BLEU,LKA}_1982-04-05.xml, {MEX,NLD}_1998-05-13.xml

(D2) Category 2 (*EN_PDF_SEMI*): {ALB,LTU}_2007-03-28.xml, {ASEAN,AUS,NZL}_2009-02-27.xml, {EU,UKR}_2014-06-27.xml

(D3) Category 3 (*GOODMT_MIXED_SEMI*): {ARE,SYR}_1997-11-26.xml, {ARG,CRI}_1997-05-21.xml, {BHR,FRA}_2004-02-24.xml

(D4) Category 4 (*BADMT_MIXED_SEMI*): {GIN,TUN}_1990-11-18.xml

B.2 Scripts

Scripts were written in Python 2.7.12 and `TensorFlow` 1.1.0.

(S1) Article extraction and preprocessing: `article_extraction_preprocessing.py`

(S2) Title normalization: `title_normalization.py`

(S3) Comparison of four strategies to compose document embeddings: `comparison_doc_embeddings.py`

(S4) Retraining of word embeddings with the SNIS corpus: `retraining_w2v_snis.py`

(S5) Computation of article embeddings: `article_embeddings.py`

(S6) K-means clustering: `kmeans.py`

(S7) Supervised learning in `scikit-learn`: `classifiers.py`

(S8) CNN classifier: `data_helpers.py, text_cnn.py, traincnn.py, evalcnn.py`

B.3 Annotations

(A1) 100 instances of the titled articles: `100titled_articles.xlsx`

(A2) 100 instances of the untitled articles: `100untitled_articles.xlsx`

C Definitions of Ten Topics in IIAs

Sentences in the definitions of topics are mostly literally selected from Salacuse [2015, Chapter 5: The General Structure of Investment Treaties, 141-154]. As certain topics are only briefly discussed in Chapter 5, we also consulted the other chapters to generate comprehensive definitions. For topic 0, additional informative sentences have been taken from ibid., Chapter 8: Investment Promotion, Admission, and Establishment, 8.1 State Sovereignty and Foreign Investment, 213-214. The definition of topic 2 was created by ourselves based the results of clustering, see *Number of clusters* in Section 5.2. Some sentences in topic 4 are taken from ibid., Chapter 14: Investment Treaty Exceptions, Modifications, and Terminations, 14.1 The Tensions of Investment Treaties, 376. The definition of "losses from armed conflict or internal disorder" under topic 8 has been taken from ibid., Chapter 13: Other Treatment Standards, 13.4 Compensation of Losses Due to War, Revolution and Civil Disturbance, 367-368. The definition of topic 9 was taken from ibid., Chapter 1: A Global Regime for Investment, 1.4 The Application of Regime Theory to Investment Treaties, 10. We manually proved the cohesion and coherence of sentences and made only minimal changes to connectives and determiners, to create internally coherent definitions of each topic.

0 Conditions for the entry of foreign investment and investors Virtually all investment treaties deal with the entry or establishment of investments emanating from treaty partners. Recognizing the importance of investment, and particularly foreign investment, to the economic prosperity and well-being of their populations while also being conscious of the potential costs that certain types of investment may entail, all states have exercised their sovereign authority to develop policies and laws to govern the admission and operation of foreign investment. This legal regime defines the types of investments that foreigners are permitted to make, the incentives they may receive, the controls to which they are subject, and the governmental agencies that have special responsibility for promoting and regulating foreign investment. One of the aims of the investment treaty movement has been to reduce internal barriers fo foreign investment, particularly through treaty provisions on investment promotion, admission and establishment.

1 Definitions and scope of application In defining the nature of covered investments, most investment treaties take four basic considerations into account: (1) the form of the investment; (2) the area of the investment's economic activity; (3) the time when the investment is made; and (4) the investor's connection with the other contracting state.

2 Others (other political, economical, cultural, technological and scientific cooperation) Other articles that are included in the IIAs on cooperation in other areas (apart from that on investment) such as agriculture, fishery, human rights, tourism, etc.

3 Operational and other conditions Investment treaties sometimes provide treatment standards

with respect to certain operational conditions, such as the investor's right to enter the country, employ foreign nationals, and be free of performance requirements. One of the most important conditions, of course, is the ability of the investor's employees to enter the host country freely and manage and operate the investment. Most investment treaties do not grant the investor an automatic right to enter and stay in a host country. Certain BITs, for example, provide that each contracting party will give 'sympathetic consideration' to applications for entry.

4 Treaty entry, exceptions, modifications and terminations Because of the great diversity of national policies and situations, it is natural that in negotiating investment treaties individual states seek to introduce exceptions to their investment treaties' obligations in order to take into account national policies and situations. Thus, most investment treaties have provisions that carve out exceptions to the general standards of treatment that they seek to apply to investments between the two countries. Investors considering a particular investment should understand the scope and force of such treaty exceptions. No treaty is ever permanent and unchanging. Thus, most international agreements, including investment treaties, contain provisions describing the process for terminating a treaty and in a few instances for modifying treaty provisions. A state has three basic devices to mediate the tensions created by investment treaty practice. The first, which is employed as part of the negotiating process, is to create specific exceptions in the treaty to assure a host state sufficient altitude of action for the future. The other two, which are invoked after the investment treaty enters into effect, are for a state to modify the treaty provisions by agreement with other contracting parties or terminate participation in the treaty and thus end its international investment obligations.

5 Dispute settlement A fundamental, practical question, of course, is whether countries actually respect their treaty commitments and, if not, whether an injured investor has effective legal redress against a host country's treaty violations. For foreign investors and their governments, one of the great deficiencies of customary international law has been its lack of effective and binding mechanisms to resolve investment disputes. One aim of the investment treaty movement has been to remedy this situation. Most investment treaties, provide for two distinct dispute settlement mechanisms: one for disputes between the two contracting states and another for disputes between a host country and an aggrieved foreign investor. Together, this results in a relatively effective system of foreign investment protection. It is also to be noted that decisions of arbitrary tribunals, although unfortunately not always made public, tend to be lengthy, reasoned, and scholarly decisions that form part of the jurisprudence of this emerging international investment law and also solidify and give force to investment treaty provisions.

6 General standards of treatment of foreign investments and investors Investment treaties stipulate the standard of treatment a host country must accord to a foreign investment in two respects. They define certain general standards of treatment and also state specific standards for particular matters such as monetary transfers, the seizure of investment property, the employment of foreign personnel, and the resolution of disputes with the host government. In addition, some general standards, such as guarantees of full protection and security or fair and equitable treatment, are absolute in nature. Others, such as national treatment or most-favored-nation treatment, are considered contingent or relative because their application depends on the treatment accorded by the state to other investors. One may identify six general standards of treatment: (a) fair and equitable treatment; (b) full protection and security; (c) protection from unreasonable or discriminatory measures; (d) treatment no less than that accorded by international law; (e) the requirement to respect obligations made to investors and investments; and (f) national and/or most-favored-nation treatment. An individual investment treaty may provide for some or all of these treatment standards.

7 Monetary transfers For any foreign investment project, the ability to repatriate income and capital, to pay foreign obligations in another currency, and to purchase raw materials and spare parts from abroad is crucial to a project's success. 'Transfer' has also become a term of art in investment treaties and basically means 'monetary transfers'. The monetary transfer provisions of most investment treaties deal with five basic issues: (1) the general nature of

the investor's rights to make monetary transfers; (2) the types of payments that are covered by the right to make transfers; (3) the currency with which the payment may be made; (4) the applicable exchange rate; and (5) the time within which the host country must allow the investor to make transfers.

8 Compensation (expropriation and dispossession/losses from armed conflict or internal disorder) One of the primary functions of any investment treaty is to protect foreign investments against nationalization, expropriation, and other forms of interference with property rights by host country governmental authorities. Despite opposition by some developing nations in multilateral forums, virtually all investment treaties adopt some variation of the traditional western view of international law that a state may not expropriate an alien's property except: (1) for a public purpose; (2) in a non-discriminatory manner; (3) upon payment of just compensation; and, in most instances, (4) with provision for some form of judicial review. Many investment treaties also deal with investment losses due to armed conflict or internal disorder within the host country. They do not, however, normally establish an absolute right to compensation in such cases. Thus, if an investor sustains a loss due to war, civil disturbance, revolution, or natural calamities, the host state will not be liable for compensation unless it failed to exercise due diligence to protect the investor. A state acts with due diligence when it makes reasonable efforts and uses the forces at its command, such as the army and the police, to protect the investor's interests to the extent practicable and feasible.

9 International governance and regime in IIAs Regime elements on international governance: principles, norms, rules, decision-making processes.

Universität Zürich[UZH]

Philosophische Fakultät
Studiendekanat

Universität Zürich
Philosophische Fakultät
Studiendekanat
Rämistrasse 69
CH-8001 Zürich
www.phil.uzh.ch

Selbstständigkeitserklärung

Hiermit erkläre ich, dass die Masterarbeit von mir selbst ohne unerlaubte Beihilfe verfasst worden ist und ich die Grundsätze wissenschaftlicher Redlichkeit einhalte (vgl. dazu: http://www.uzh.ch/de/studies/teaching/plagiate.html).

Niederweningen 22.06.2017 .....

Ort und Datum Unterschrift