A Granular-based Approach for Semisupervised Web Information Labeling

by

Cenker Sengoz

A Dissertation Submitted in Partial Fulfillment of the
Requirements for the Degree of

Master of Science

in the Department of Applied Computer Science

A Granular-based Approach for Semisupervised Web Information Labeling

by

Cenker Sengoz

Supervisory Committee

Dr. S. Ramanna, Supervisor
(Department of Applied Computer Science)

Dr. C.J. Henry, Departmental Member
(Department of Applied Computer Science)

Dr. J.T. Yao, External Member
(Department of Computer Science, University of Regina)

Supervisory Committee

Dr. S. Ramanna, Supervisor
(Department of Applied Computer Science)

Dr. C.J. Henry, Departmental Member
(Department of Applied Computer Science)

Dr. J.T. Yao, External Member
(Department of Computer Science, University of Regina)

ABSTRACT

A key issue when mining web information is the labeling problem: data is abundant on the web but is unlabelled. In this thesis, we address this problem by proposing i) a novel theoretical granular model that structures categorical noun phrase instances as well as semantically related noun phrase pairs from a given corpus representing unstructured web pages with a variant of Tolerance Rough Sets Model (TRSM), ii) a semi-supervised learning algorithm called Tolerant Pattern Learner (TPL) that labels categorical instances as well as relations. TRSM has so far been successfully employed for document retrieval and classification, but not for learning categorical and relational phrases. We use the ontological information from the Never Ending Language Learner (Nell) system. We compared the performance of our algorithm with Coupled Bayesian Sets (CBS) and Coupled Pattern Learner (CPL) algorithms for categorical and relational labeling, respectively. Experimental results suggest that TPL can achieve comparable performance with CBS and CPL in terms of precision.

Keywords: Semi-supervised learning, Web information labeling, Never Ending Language Learner, Granular Computing, Tolerance Rough Sets.

Contents

List of Tables

List of Figures

ACKNOWLEDGEMENTS

First of all, I'd like to thank Prof. Sheela Ramanna who has been more than a supervisor to me. From the beginning to the end, she has been a great mentor; always very supportive, caring and enabling in my high and low moments. She gave me the opportunity to come and study in this wonderful country. I'm very grateful to her for everything and I consider it a privilege to be one of her students. I'm also thankful for the Natural Sciences and Engineering Research Council of Canada (NSERC) Discovery grants which have made this study financially possible on my end. I'd further like to acknowledge all the support and guidance I had by Prof. Estevam Hruschka, Prof. James F. Peters, Dr. Andrew Carlson and Mr. Saurabh Verma who all contributed this research in many ways. I pass my gratitudes to my former instructors Dr. Christopher J. Henry, Dr. Pradeep Atrey, Prof. Simon Liao, Prof. Yangjun Chen and to all the other valued members of the Applied Computer Science Department including Mr. James Deng and Ms. Connie Arnhold. Special thanks to our Faculty Dean Prof. Mavis Reimer, the Graduate Studies Officers Mr. Eric Benson, Ms. Deanna England, Ms. Dagmawit Habtemariam, the Faculty of Graduate Studies and the U of W family. I appreciate the presence and the support of all my friends and colleagues including Ankita, Fatima, Shreelatha, Violet, Kanwar, JingJing, Shy, Harmeet, Asish, Henry, Leo, Ashmeet, Bakul, Lipan, Abukari, Orlando, Aditya, Aqeel, Arshia, Manish, Gagandeep, Varun, Başar, Efe, Sercan, Yusuf, Barış, Canan, Hazim, Anna, Katya, Dari and especially Kasun who has always looked after me like the elder brother I never had. I'd also like to thank the examining committee including Prof. Ramanna, Dr. Henry and Prof. JingTao Yao for all the time and effort they will donate examining this work. Last but not least, I'm grateful to my dear parents Mrs. Elif & Mr. Tamer Şengöz and my brother Türker who have always been there for me. Without their support, I could not have made it this far in life. Thanks to everyone else whom I haven't named that has been with me throughout this journey. Thank you Manitoba and Canada for being so wonderful and welcoming. I love you all!

Cenker Şengöz

DEDICATION

Anneme, babama ve kardeşime...

To my mother, my father and my brother...

Chapter 1

Introduction

According to *Internet World Stats*[1], in December 1995, there were 16 million Internet users worldwide. This number is estimated to be 2.9 billion for March 2014, depicting a growth from 0.4% to 40.9% of the global population. Within just two decades, Internet has become something so vast and fundamental that it has spawned a new age of knowledge. With an ever increasing amount of information that is available on the Web, there is also an emerging need for automated systems to extract and structure information on the web. Much research has been taking place for extracting relational facts from both structured and unstructured text. Web information extraction (IE) systems such as YAGO [36], KnowItAll [8], TextRunner [1], and Nell [4] gather entities and factual relations between entities from Web sources. These systems employ classical machine learning as well as statistical classifiers.

A major issue when learning from the Web is the labeling problem: data is abundant on the web but it is unlabelled. This condition renders supervised methods, which rely on labelled data, inapplicable for most web information extraction problems. In accordance, unsupervised methods are used for the most cases. Another option is the semi-supervised learning approach and it is adopted by the *Never Ending Language Learner (Nell)* [4]. Operational since 2010, Nell is a computer agent that iteratively extracts and organizes relevant information from the Internet to grow and maintain a knowledge base of facts. The facts in question are represented by two means: categorical instances e.g. *City*(Winnipeg) and semantically related pairs e.g. *City-In*(Winnipeg, Canada) for which the categories and relations are defined in advance. The core component of Nell, called *Coupled Pattern Learner (CPL)* [3], is a

[1] http://www.internetworldstats.com/emarketing.htm

semi-supervised algorithm extracting *noun phrases* that appear in free text form. To do so, it uses *contextual extraction patterns*, a sequence of words providing a context for the instance by preceding, succeeding, or surrounding the noun of interest (e.g. "*industrialized countries like arg*"). These nouns and contexts are extracted from web text by means of natural language processing along with their co-occurrence information. A more recent treatment to Nell's categorical information extraction problem was made by Verma et. al [39] in 2012. They extract categorical noun phrase instances by simultaneously learning independent classifiers in a new approach named *Coupled Bayesian Sets (CBS)* algorithm. CBS outperforms CPL after 10 iterations rendering it a good possible alternative for the CPL component of Nell. Nell became our default Web IE system because of access to crucial data set that was necessary for our experiments and for comparisons with CBS and CPL methods.

Rough Set theory has become one of the essential foundations of granular computing [22]. Granular computing is an umbrella term to cover any theories, methodologies, techniques, and tools that make use of information granules in complex problem solving [46, 25]. A granule is a clump of objects (points) in the universe of discourse drawn together by indistinguishability, similarity, proximity, or functionality [47]. Granulation leads to information compression/summarization. Reasoning with granular models is particularly useful in machine learning problems with incomplete information or unsharp class boundaries. Granulation with classical rough sets where the concept of indiscernibility (similarity) formed by an equivalence relation is plausible for text mining, but too restrictive. Instead a tolerance rough sets based model (TRSM) [21, 29, 16, 27, 41, 26] that admits overlapping granules and specifically generalized tolerance approximation spaces [34] has been proposed. TRSM is a granular model which has so far been used for term-described document representation for the task of document classification and clustering [11, 12, 20, 40].

1.1 Problem Definition

In this thesis, we consider the type of web information labeling problem addressed by the Never Ending Language Learner system henceforth termed as the "Nell problem", populating:

1. categorical noun phrase instances (e.g. *Sport(Ice Hockey)*)

2. relational noun phrase pairs (e.g. *Popular-Sport-Of(Canada, Ice Hockey)*)

by using the contextual extraction patterns and their co-occurrence statistics with noun phrases (for categories) and with noun phrase pairs (for relations) from a data corpus of web documents. Categories and relations are predefined. Noun phrase literals, contextual patterns and the co-occurrence data are assumed to be tokenized and retrieved from web corpora in advance.

1.2 Proposed Approach

We address the problem by using a granular model based on tolerance rough sets. We propose and examine

i) a granular model that structures categorical noun phrase instances as well as related noun phrase pairs from a given corpus representing unstructured web pages,

ii) a semi-supervised learning algorithm we call Tolerant Pattern Learner (TPL) that labels categorical instances as well as relations.

To the best of our knowledge, this work is the first attempt in the literature at using a granular-based approach to labeling context-described categorical and relational noun phrases. We have been inspired by CBS and CPL in terms of the semi-supervised approach we take for the problem. We have also been inspired by a tolerance form of similarity rooted in granular computing. TRSM has been successfully employed for document retrieval and classification yet its suitability for this particular domain (semi-supervised labeling of noun phrases and noun phrase pairs as categorical and relational instances) was to be explored.

1.3 Contributions

Our contributions [31, 32] are threefold:

- We propose a novel theoretical granular model for context-described noun and relational phrase learning problem.
- We provide a practical implementation of our model as a semi-supervised algorithm for categorical and relational information labeling.

- We present empirical evidence on the performance of our proposed solution, by comparing our approach to the benchmark algorithms CBS and CPL for the categorical and relational learning problems, respectively.

1.4 Thesis Layout

The rest of this thesis organized as follows:

Chapter 2 provides a background for web information extraction (IE) along with some state-of-the-art domain-specific and domain-independent systems.

Chapter 3 introduces the semi-supervised learning paradigm and the never ending language learning (Nell) problem. It also describes the two semi-supervised algorithms addressing the Nell problem, Coupled Bayesian Sets (CBS) and Coupled Pattern Learner (CPL), which we used as benchmarks.

Chapter 4 discusses rough sets, tolerance approximation spaces and tolerance rough sets model (TRSM) along with its applications in document clustering and text clustering. All of these serve as basis for the theoretical framework of this thesis.

Chapter 5 provides the theoretical framework for the granular model we propose for the categorical/relational learning problem. It also covers the proposed Tolerant Pattern Learner algorithms for learning categorical and relational facts.

Chapter 6 gives the implementation details, experiments, results and discussion on TPL and the described granular model.

Chapter 7 concludes the thesis, summarizes the work done and provides possible future research directions.

Chapter 2

Web Information Extraction

Information extraction (IE) is defined as the task of automatically extracting structured information from unstructured and/or semi-structured machine-readable documents [14]. For web information extraction, the documents in question are web pages which may contain information as free text paragraphs or embedded in structures like HTML lists. This extraction activity may involve steps from locating the information, abstracting it from its surrounding by means of natural language processing and structuring it within the desired context by means of machine learning. Figure 2.1 shows an example of this transformation. In this particular example, sentences are decomposed into different subcomponents all of which are translated according to their role in the sentence to obtain an ordered list of different categories.

Within the last decade, significant effort has been dedicated by the machine learning community to extract information in various forms of factual data from web-based corpora. Some of these systems were designed to operate on specific domains or websites whereas the others are intended to be more generic. Rest of this section discusses some web information extraction systems which drew attention in the recent past.

2.1 Domain-Independent Projects

2.1.1 KnowItAll

In 2005, Etzioni et al. [8] proposed *KnowItAll*, an unsupervised web information extraction system that can automatically extract facts from the web. The facts can be in form of unary category instances or n-ary relation tuples, where the predicates are defined in advance. Within a two-step process, KnowItAll first uses a small

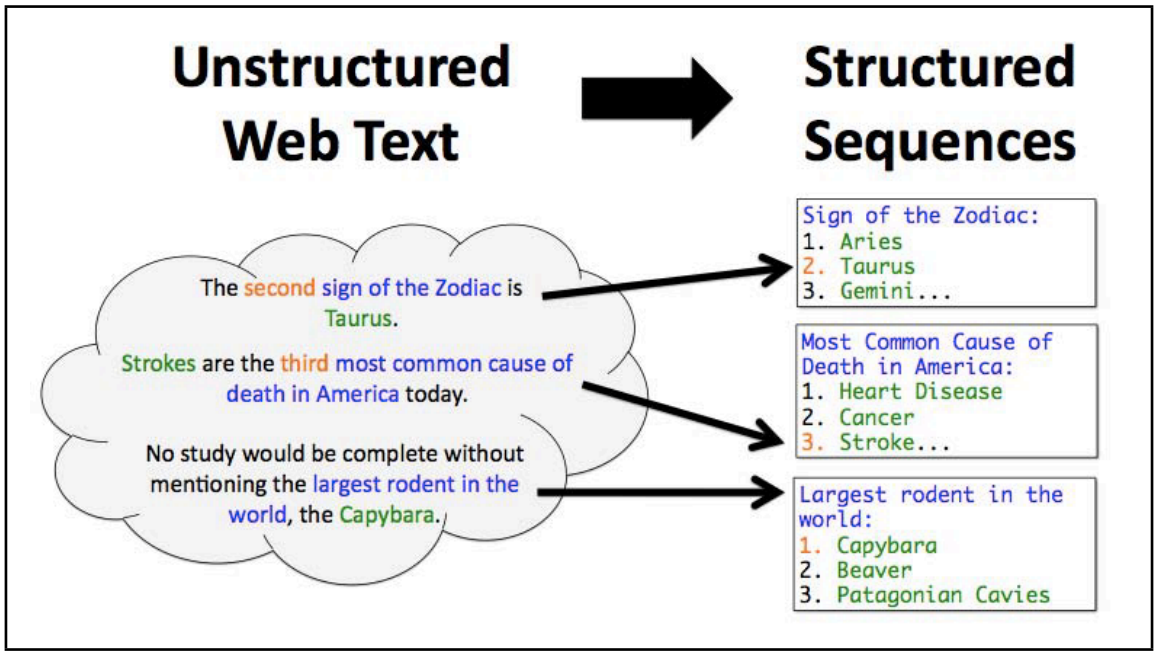


Figure 2.1: Structured information extraction from unstructured text [30]

number of generic extraction patterns as well as part-of-speech tagging to determine the candidate facts. Then it calculates a probability of correctness for candidate facts. The assessor module bases its probability on search engine hit counts used to compute the mutual information between the extracted instance of a class and a set of automatically generated discriminator phrases associated with that class [8].

2.1.2 TextRunner

Banko et al. [1] introduced a domain-independent unsupervised information extraction paradigm called *Open IE* along with *TextRunner*, a system that implements Open IE on a large scale to extract relational facts. Basically, Open IE eliminates the need for a pre-defined lexicon for asserting candidate arguments. The input of *TextRunner* is a web corpus and the output is a set of extracted relations. This approach espouses a single-pass relation discovery. It uses part-of-speech tagging by going through the corpus sentence-by-sentence and determining the candidate tuples as well as relations. The candidates are then assigned a probability by a self-supervised learner that employs a Naive Bayes classifier, which are eventually refined and indexed to support efficient extraction and exploration via user queries. The authors reported that TextRunner was able to match the recall of KnowItAll and it could achieve a

better precision value [1].

2.1.3 ReVerb

ReVerb [9] is another information extractor that utilizes the Open IE paradigm. It is composed of two modules, a relation extractor and an argument extractor to acquire the relations and the related noun phrases, respectively. Just like the previous models, part-of-speech tagging is at the core of this extraction process. ReVerb imposes a syntactic constraint which ensures a relation phrase follows a white-listed part of speech tag pattern as well as a lexical constraint which filters out the over-specified phrases to limit the extraction of incoherent and uninformative word sequences as relations. ReVerb is shown to outperform TextRunner and another state-of-the-art Open IE system WOE discussed in Section 2.2.2 in terms of precision and recall [9].

2.1.4 FreeBase

In 2007, a company called Metaweb released *FreeBase* [1], an on-line structured knowledge base collaboratively formed and maintained by its community [17]. The data it uses are collected across the world wide web, including but not limited to, websites such as Wikipedia and MusicBrainz as well as community contributions. Its ontology is composed of domains (e.g. *TV*), topics (e.g. *TV Program*) and properties (e.g. *Number of Episodes*). It uses a non-hierarchical graph model depicting entities as nodes and their semantic relations as links. The owning company was acquired by Google in 2010 [18]. Details regarding the underlying technology of the system are proprietary, as of today.

2.2 Domain-Specific Projects

2.2.1 YAGO

YAGO (Yet Another Great Ontology) is said to be a lightweight and extensible knowledge base [36]. It adopts an entity-relationship model; maintaining both taxonomic (is-a) and non-taxonomic (has-a) relations between entities. YAGO relies entirely on Wikipedia and WordNet (a lexical database for English). Entities are extracted from these systems for predefined relations. YAGO uses a variety of heuristics to extract

[1] http://www.freebase.com

knowledge. For example, it leverages the category pages from Wikipedia, which are lists of articles belonging to a given category. *"Zinedine Zidane"* is present in the page *"List of international French footballers"*. This is used to create the candidate entity *Zidane* and to build the candidate relationships *is-A(Zidane, footballer)*, and *citizen-of(Zidane, France)*. The authors state that their empirical evaluation of fact correctness shows an accuracy of about 95% [36].

2.2.2 WOE

WOE (Wikipedia-based Open Extractor) [45] is another instance of the OpenIE systems like TextRunner. As the name implies, it uses Wikipedia as its source. WOE has a layered architecture consisting of three parts: preprocessor, matcher and learner. Processor splits pages to sentences, parses annotations via natural language processing and compiles synonym tokens. Matcher creates the training data by matching the attribute values in Wikipedia infoboxes (small boxes on the top right corner of a typical article) with the corresponding article sentences using heuristics. Ultimately, learner forms two independent extractors, WOE^{parse} using dependency parse-trees and WOE^{pos} using part-of-speech tags. The former one is shown to run at the same speed as TextRunner with an F-Measure between 15% and 34% better. The latter one yields 79 % and 90 % improved F-measure against TextRunner with the cost of 30x slower execution due to parsing, the authors conclude [45].

2.2.3 TWICAL

TWICAL [28] is an open-domain event extraction and categorization system for Twitter status messages. Events to be extracted are represented as 4-tuple e.g. *(entity = 'iPhone', event phrase = 'announcement', calendar date = '10/4/11', event type = 'ProductLaunch')*. Given a raw stream of tweets, first the tweets are POS tagged, then named entities and event phrases are extracted, temporal expressions resolved and the extraction events are categorized into types [28]. The approach uses latent variable models to uncover the set of types matching the data. By leveraging large volumes of unlabelled data, it outperformed a supervised baseline by 14% increase in maximum F1 score [28].

2.2.4 DBpedia

DBpedia [2] [15] is a structured knowledge base manufactured on top of the Wikipedia project. Like WOE and YAGO, DBpedia also relies on the semi-structured information on that website, such as infoboxes, article categories, annotations and links. Factual information is extracted and organized from possibly multiple Wikipedia articles into a uniform data set, making the content easier to retrieve. DBpedia has an SQL-like interface called SPARQL supporting complicated queries (e.g. *"All German musicians that were born in Berlin in the 19th century"*). A stable version of DBpedia has been recently released in 2014.

[2] http://dbpedia.org/

Chapter 3

Semi-supervised Learning and the Nell Problem

3.1 What is Semi-supervised Learning?

Semi-supervised learning (SSL) is a machine learning paradigm that combines supervised (SL) and unsupervised learning (USL). Let us first elaborate these two constituting paradigms of SSL.

Supervised methods rely entirely on labelled i.e. "supervised" data to fulfill their tasks, where the task can be statistical classification or information extraction. Typically, an entity x is represented as a vector $x = (x_0, ...x_n)$ in an Euclidean space R^n where each dimension corresponds to a different aspect, characteristic or attribute of the entity. Then the goal is to find the function f that provides a mapping from the entity x to the output class (label) y [6]. Supervised methods assume an independent and identical distribution (i.i.d) over the entity-label pairs. The system is trained (i.e. the function f is calibrated) via the labelled training examples, and the resulting function is used as a classifier to determine the labels for the unlabelled examples. This process is summarized in Figure 3.1. On the left, objects with '?' are initially unlabelled; triangles and rectangles are the training examples of their respective classes. On the right, all the objects are classified by the classifier f.

Unsupervised learning is the alternative when there are no labelled training examples available. The task is to reveal the underlying structure of the data from the point of interest, by using unlabelled training points independently and identically sampled from a shared distribution. For this paradigm, the problem in hand is essen-

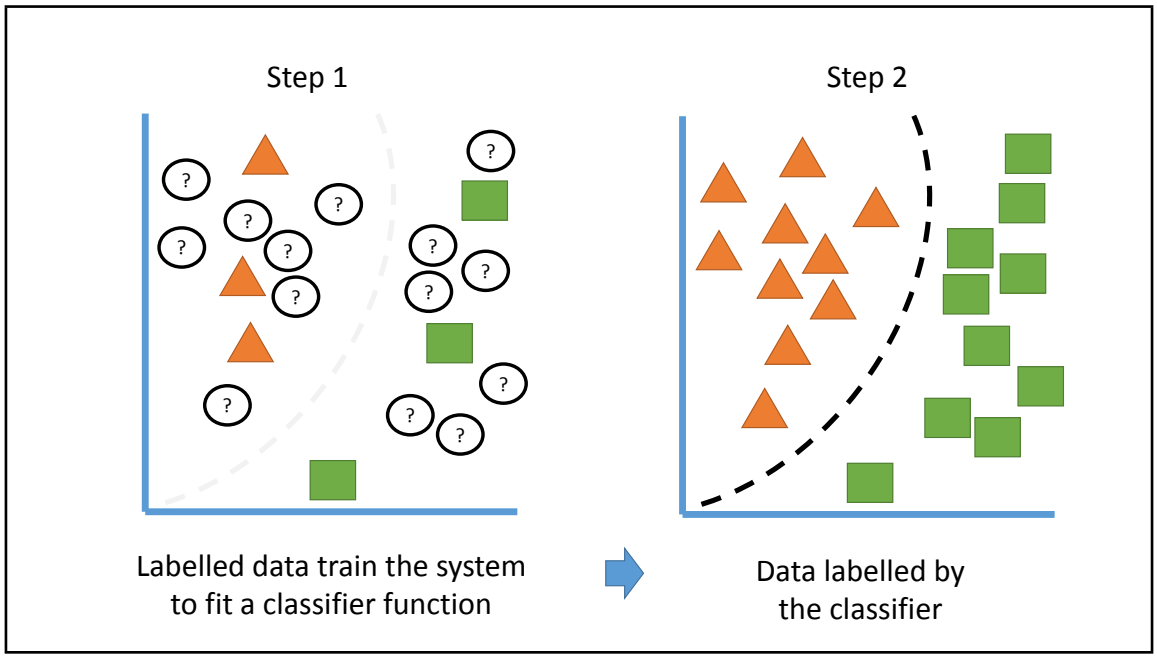


Figure 3.1: Supervised learning

tially to estimate the density which is likely to have generated the data distribution in question [6]. In accordance, data points are clumped along with the points in their neighbourhood, forming clusters of objects (see Figure 3.2).

As one might expect, both approaches have their strengths and weaknesses. The supervised framework provides a better chance to properly constrain the learning process and thus offers a better performance. Unfortunately, it suffers from scalability if the data in hand is large: labelled examples become expensive to obtain and when they are limited in number, they simply cannot accurately sample and represent the entire data set. Unsupervised learning is more easily scaled for larger sets, yet it is less reliable.

Semi-supervised learning is the result of an attempt to leverage the stronger sides and to suppress the drawbacks of both methods. It incorporates both labelled and unlabelled data for training to adjust the classifier function. This way it becomes better scalable to larger data than SL and it tends to be more reliable than USL since human supervision is involved.

A typical course of action for SSL is iteration. The system is "bootstrapped" by using the provided labelled examples: They are used to label a first round of data set, where the labelled data constitutes exclusively high-confidence instances. Those examples are then fed-back to the system to further train it and to provide a

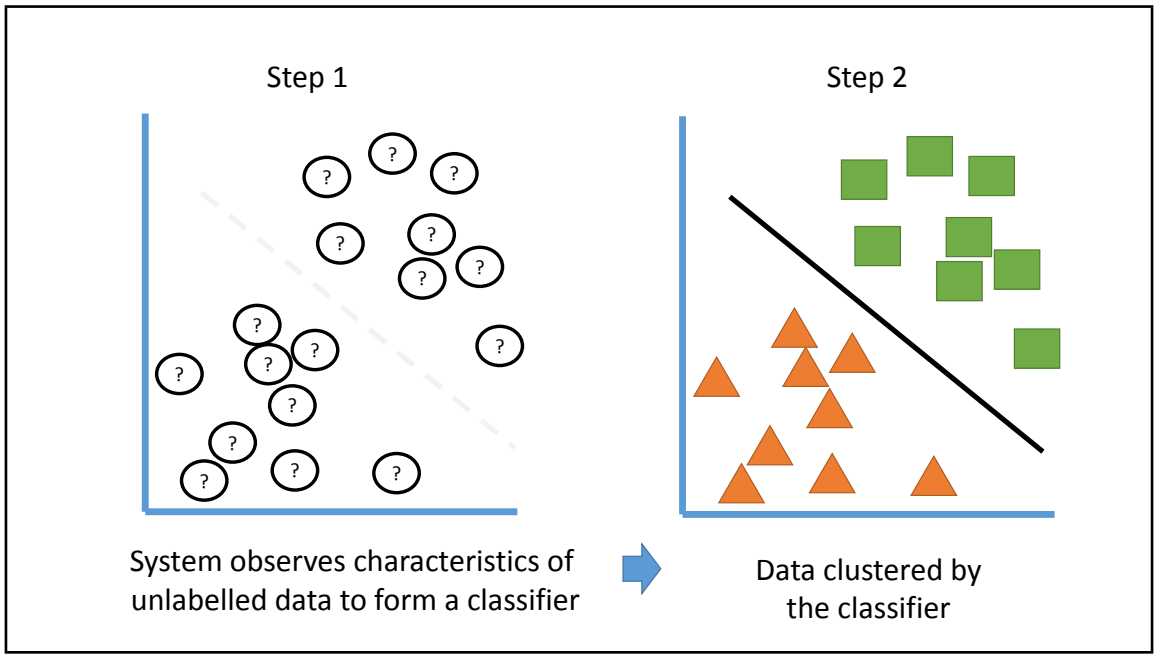


Figure 3.2: Unsupervised learning

second round of results (see Figure 3.3). This iterative process is repeated as long as desired, typically, until the classifier converges and all the elements are classified or indefinitely, depending on the problem domain.

Semi-supervised learning has found its applications in web information extraction some of which are discussed in the remainder of this chapter.

3.2 Never Ending Language Learner (Nell)

In 2010, Carlson et al. started to develop a computer system called the Never Ending Language Learner (Nell)[1] [4]. Its task is to extract and structure relevant information continuously to grow a knowledge base of facts. The facts in question are represented by two means:

- **Category** instances e.g. *City*(Winnipeg)
- **Relation** pairs e.g. *City-In*(Winnipeg, Canada)

It operates in an iterative semi-supervised fashion. The ontology is initialized by a limited number of labelled examples for every category/relation. Those examples

[1] http://rtw.ml.cmu.edu/rtw/

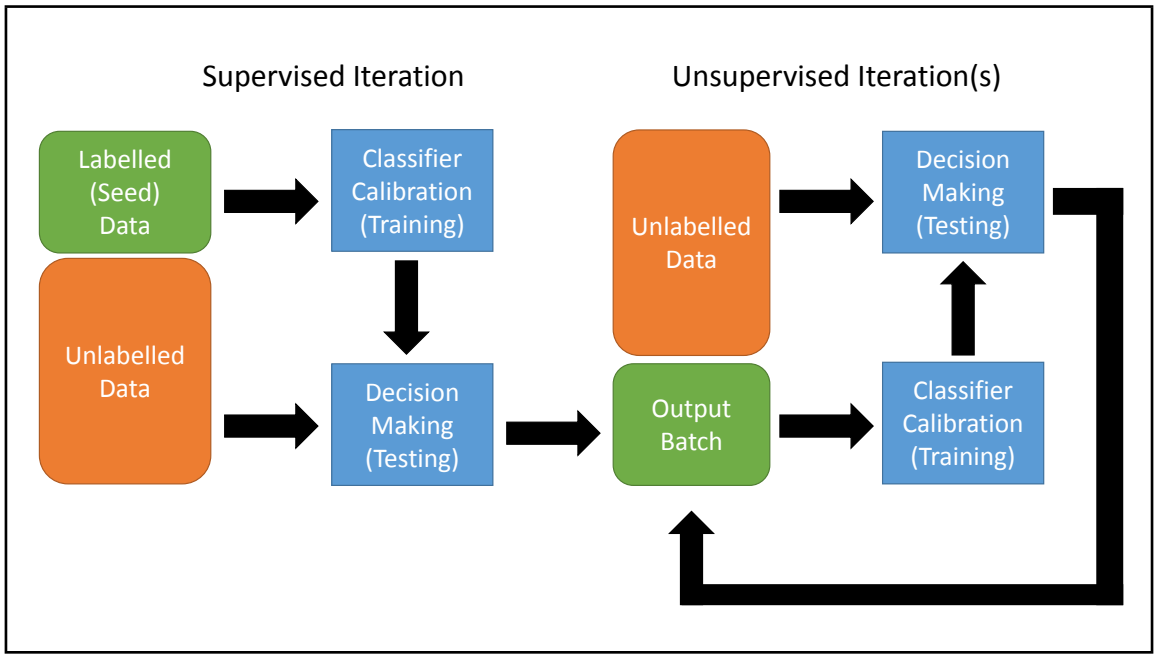


Figure 3.3: Iterative semi-supervised learning

bootstrap the system and are used to label more instances, which will then be used to label more, leading to an ever-growing knowledge base.

It has been noted that semi-supervised approaches using a small number of labelled examples together with many unlabelled examples are still unreliable as they frequently produce an internally consistent, but nevertheless, incorrect set of extractions [39]. While such semi-supervised learning methods are promising, they might exhibit low accuracy, mainly, because the limited number of initial labelled examples tends to be insufficient to reliably constrain the learning process, creating concept drift problems [7]. To overcome this issue, Carlson et al. couple the iterative training by using the following 3 constraints [3]:

1. **Output constraints**: For two functions $f_a : X \to Y_a$ and $f_b : X \to Y_b$, if there are some constraints known on values y_a and y_b for an input x, one can require f_a and f_b to satisfy this constraint. As an example, if f_a and f_b are Boolean-valued functions and $f_a(x) \to f_b(x)$, $f_b(x)$ could be constrained to have value 1 whenever $f_a(x) = 1$.

2. **Compositional constraints**: For two functions $f_1 : X_1 \to Y_1$ and $f_2 : X_1 \times X_2 \to Y_2$ there may be a constraint on valid y_1 and y_2 pairs of a given x_1 and any x_2. One can require f_1 and f_2 to satisfy this constraint. For example, f_1 could

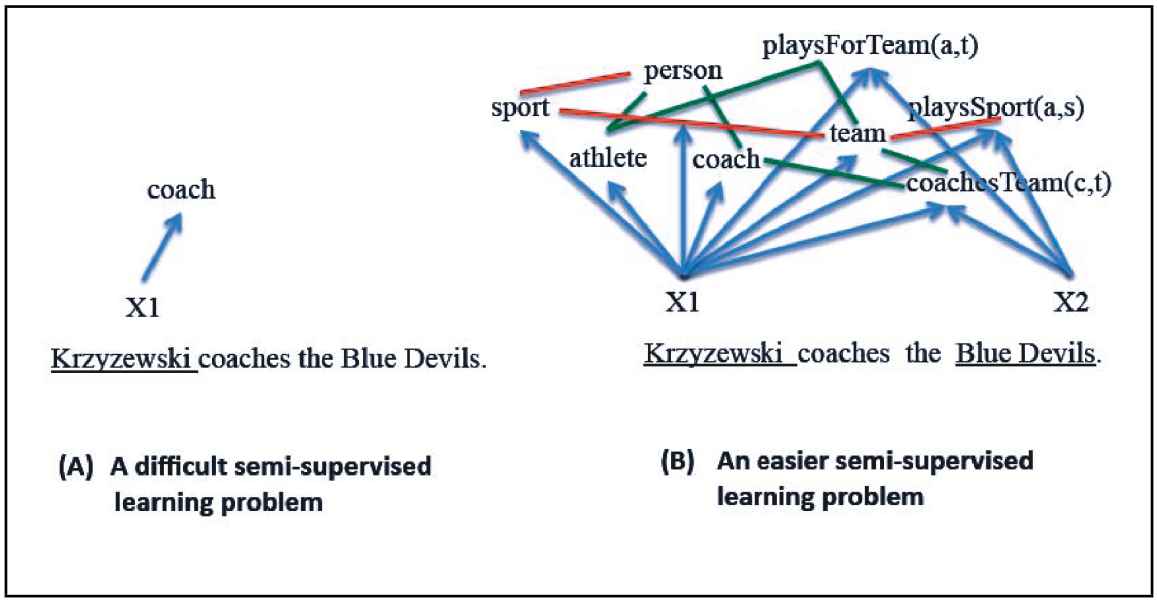


Figure 3.4: Coupling constraints in action [3]

"type check" valid first arguments of f_2 so that $\forall x_1, \forall x_2, f_2(x_1, x_2) \rightarrow f_1(x_1)$.

3. **Multi-view-agreement constraints**: For a function $f : X \rightarrow Y$, if X can be partitioned into two "views" where one writes $X = \langle X_1, X_2 \rangle$ and assumes that both X_1 and X_2 can predict Y, then one can learn $f_1 : X_1 \rightarrow Y$ and $f_2 : X_2 \rightarrow Y$ and constrain them to agree. For example, Y could be a set of possible categories for a web page, X_1 could represent the words in a page, and X_2 could represent words in hyper-links pointing to that page.

These constraints are used to cross-check the characteristics of candidates and their relations, which provides additional justification before committing to an assertion. Figure 3.4 illustrates this procedure.

As a comprehensive IE system, Nell relies on a number of subcomponents each of which are designed to work on a complementary basis [3]:

- **Coupled Pattern Learner (CPL)** is the core component of Nell whose task is free text extraction. This will be discussed in detail in the following subsection.
- **Coupled SEAL (CSEAL)** is a set expansion algorithm that extracts instances from semi-structured documents. It exploits the signatures of the wrapping HTML tags to locate the desired content and extract them with the help of the coupling constraints discussed above.

- **Meta-Bootstrap Learner (MBL)** is the governing algorithm that merges the results of the other two. It enforces the mutual exclusion and type-checking constraints on the results and commits the ones which do not violate the constraints.

3.2.1 Coupled Pattern Learner (CPL) Algorithm

Coupled Pattern Learner (CPL) [3] uses what are called *contextual patterns* to detect and extract *noun phrase* instances for the knowledge base. Those patterns are sequences of words e.g. "sports such as *arg*" or "*arg1* is the president of *arg2*" providing a context for a noun phrase argument. The main idea is, noun phrases that are likely to belong to a particular category/relation are also likely to co-occur frequently with the patterns associated to that category. Accordingly, the co-occurrence information between noun phrases and contextual patterns is what CPL relies on for learning.

CPL is actually a two-way algorithm: Contexts are used to find the nouns and nouns are used to find the contexts. This is done in a sequential and iterative manner. In the noun learning mode, trusted contexts assigned to the category of interest behave as features and are used to label more nouns. In the context learning mode, the roles reverse and the nouns are used as features to acquire more contexts. In either mode, the information used is the noun-context co-occurrence statistics.

The overall flow of the CPL algorithm is summarized in Algorithm 1. There are four fundamental steps which are repeated for learning noun phrases and the contextual patterns. An analogous process is adopted for learning relational pairs as well. [3]:

1. *Candidate Extraction:* In each iteration, CPL uses the recently promoted patterns to extract candidate instances. Particularly, it selects the 1000 candidates which occur with the most number of patterns from the previous round. In the first round, the seed patterns are used.

2. *Candidate Filtering via Coupling:* Mutual exclusion and type-checking constraints are enforced to filter out the low-confidence candidates. This is done in a soft manner i.e. a candidate is not immediately rejected after the constraint is violated. It is considered for further processing as long as the number of times it co-occurs with a promoted pattern is at least 3 times more than the number of times it co-occurs with the patterns of mutually excluded predicates. According

Algorithm 1: Coupled Pattern Learner (CPL) [3]

Input : An ontology $\mathcal{O}$, and text corpus C
Output: Trusted instances/contextual patterns for each predicate

```
for i = 1, 2, ..., ∞ do
    for each predicate p ∈ O do
        EXTRACT new candidate instances/contextual patterns using recently
        promoted patterns/instances;
        FILTER candidates that violate coupling;
        RANK candidate instances/patterns;
        PROMOTE top candidates;
```

to the authors, this soft approach is much more tolerant of the inevitable noise in web text as well as ambiguous noun phrases than a hard constraint [3].

3. *Ranking:* The candidate patterns are ranked by

$$Precision(p) = \frac{\sum_{i \in I} count(i, p)}{count(p)} \tag{3.1}$$

where p is the pattern, I is the promoted instance set for the target predicate, $count(i, p)$ is the co-occurrence value of i and p, and $count(p)$ is the total hit count for the pattern in the text corpus.

4. *Promotion:* CPL promotes the top ranked candidates, provided that at most 100 instances and 5 patterns are promoted per iteration.

The performance of the CPL algorithm was empirically tested and justified [3]. A text corpus derived from 200 million web pages was used to create a data set accommodating noun phrase contextual pattern co-occurrence statistics. To elaborate some details on how this data set is prepared from the web pages, the authors [3] first parsed the HTML web documents and filtered out the pages which contain non-English or adult content using a stop-word-ratio threshold and a bad-word list. Then, the resulting collection of "useful" web pages were treated by a third party software called OpenNLP[1]. This software was used to decompose paragraphs into sentences, tokenize the sentences into parts of speech and tag each token. Then, the sentences with tagged tokens were processed to extract the candidate nouns and contexts, along with their hit counts and their co-occurrence data. This is the extent to which this

[1] http://opennlp.sourceforge.net

Table 3.1: Precision values (%) of CPL for selected categories [3]

Category	Athlete	BoardGame	City	Country	Hobby
Precision	87	80	97	57	77
Category	Politician	Reptile	Sport	Vehicle	*Average (All 44)*
Precision	80	95	77	67	*78*

Table 3.2: Precision values (%) of CPL for selected relations [3]

Relation	AthletePlaysForTeam	CityLocatedInCountry
Precision	100	93
Relation	CompanyIsInEconomicSector	TeamHasHomeStadium
Precision	93	100
Relation	StateHasCapitalCity	*Average (All 20)*
Precision	60	*89*

process is elaborated in the related work [3, 4] and further details including the execution times and the hardware used have been withheld.

The ontology was populated with instances for 44 categories and 20 relations after 10 iterations. The precision measure was calculated by sampling 30 promoted instance per predicate. For this experiment, the average precision values of 78% and 89% for categories and relations, respectively were obtained. Nevertheless, there are also a few problematic ones with poor results (see Tables 3.1 and 3.2).

3.3 Coupled Bayesian Sets (CBS) Algorithm

Inspired by Nell, Verma and Hruschka also considered the problem of extracting categorical information from unstructured web pages. They proposed the Coupled Bayesian Sets (CBS) [39] algorithm to fulfill the same task as CPL, that is, extracting noun phrases to populate instances for category. Likewise, it follows a semi-supervised approach and it makes use of the co-occurrence statistics between noun phrases and contextual patterns.

CBS is based on the Bayesian Sets Algorithm [10]. Provided with an ontology defining categories and a small number of seed examples along with a large corpus yielding the co-occurrence information between phrases and patterns, CBS calculates a probabilistic score by using those co-occurrence statistics for every category candidate; and the top ranked ones are promoted as trusted instances for that category. The promoted instances are then used as seeds in the subsequent iterations. The

algorithm also exploits the mutual exclusion relations across categories to provide further evidence for its decisions.

Let $x = (x_{.1}, x_{.2}, .., x_{.J})$ be a binary noun phrase vector where $x_{.i} \in \{0, 1\}$ is the binary weight of contextual pattern i for that noun. Furthermore, let $q^c = (q_1^c, q_2^c, .., q_J^c)$ be the binary weight vector for category C formed by using the noun phrase vectors x_i labelled for that category:

$$q_j^c = \begin{cases} 1 & \iff \exists x_i \in C \wedge x_{ij} = 1 \\ 0 & \text{otherwise} \end{cases} \tag{3.2}$$

Then, CBS ranks every candidate instance x by the following score [39]:

$$logscore(x) = \rho + \sum_j q_j^c x_{.j} - \sum_i \sum_j q_j^i x_{.j} \tag{3.3}$$

In this equation, the first two terms constitute the Bayesian log score as proposed in [10]. The final term implements the mutual exclusion constraint. The score of x for category C is penalized if the mutex constraint is violated i.e. if a positive feature j of x also matches to a mutually exclusive category i of C. The first term ρ is defined as

$$\rho = \sum_j log(\alpha_j + \beta_j) - log(\alpha_j + \beta_j + N) + log(\tilde{\beta}_j) - log(\beta_j) \tag{3.4}$$

where

$$q_j^i = log(\tilde{\alpha_j}^i) - log(\alpha_j) - log(\tilde{\beta_j}^i) + log(\beta_j) \tag{3.5}$$

Here, N is the number of vectors (noun phrases), α and β are hyper-parameters (drawn by the Beta distribution that serves as the conjugate prior for the Bernoulli distribution x is assumed to have). $\tilde{\alpha}$ and $\tilde{\beta}$ are calculated as

$$\tilde{\alpha_j} = \alpha_j + \sum_{i=1}^{N} x_{ij} \tag{3.6}$$

$$\tilde{\beta}_j = \beta_j + N - \sum_{i=1}^{N} x_{ij} \tag{3.7}$$

Verma et al. set the hyper-parameters $\alpha = \eta * m$ and $\beta = \eta * (1 - m)$ with m being the mean vector of features across all instances and $\eta = 2$ [39].

Algorithm 2 summarizes the flow of CBS. It can be observed that this iterative

Algorithm 2: Coupled Bayesian Sets (CBS) [39]

Input : An ontology $\mathcal{O}$, and a corpus C
Output: Trusted instances for each given category

```
for i = 1, 2, ..., ∞ do
    for each category do
        extract new instances using available labelled examples;
        filter instances which violate coupling;
        rank instances using score mentioned in Eq. 3.3 ;
        promote top ranked instances;
```

Table 3.3: Precision values (%) of CBS for all categories [39]

Category	Company	Disease	KitchenItem	Person
Precision	100	100	94	100
Category	PhysicsTerm	Plant	Profession	Sociopolitics
Precision	100	100	100	48
Category	Sport	Website	Vegetable	*Average (All 11)*
Precision	97	94	83	*92*

semi-supervised algorithm exhibits significant resemblance with the CPL component of Nell. The main difference is how the candidates are compared and ranked; CPL uses a scoring mechanism based on the simplistic precision score in Eq. 3.1 whereas CBS uses the Bayesian log score in Eq. 3.3.

To evaluate its performance, Verma et al. sampled 11 categories from the knowledge base of Nell, for which they have populated instances throughout 10 iterations. The performance of their algorithm was measured using Precision@30 metric. Precision@30 is calculated as follows: all the promoted instances of a specific iteration are ranked and then the percentage of correct instances in the subset formed by the top 30 entities (in the ranked list) are calculated [39]. According to the results of their experiments, CBS was able to outperform CPL in terms of this metric, rendering it a good possible alternative for the free text extractor of Nell (see Table 3.3).

3.3.1 More Related Work on the Nell Problem

Further research involving different aspects of learning categories and relations is also taking place for the Nell problem. Krishnamurthy et al. [13] introduced vector space semantic parsing, a framework for learning compositional models for vector space

semantics. This model envisions a vectorial representation for noun phrases along with additional properties which are functions on those vectors. Wijaya et al. [43] considered populating relational instances out of individual noun phrases (noun compounds, as they refer) by splitting eligible ones into their components. They aimed to derive relation instances like *Relieved-By*(Pills, Headache) via the noun compound "headache pills". Later, they proposed a model [44] to analyze the semantic changes for words over time by inspecting the evolution of contexts that accommodate a given noun phrase. Mohamed et al. [19] extended the challenge from finding the related pairs to discovering the relations, thus, creating a dynamic ontology. Later on, Talukdar et al. [38, 37] conducted studies on maintaining the temporal order of relational pairs involving events e.g. *Acted-In*(DiCaprio, Titanic) occurring before *WonPrize*(Titanic, Oscar). Very recently, Wijaha et al. [42] introduced a contextual temporal profiling for entities to establish a temporal validity window for the records in a knowledge base.

For the most part, we have kept these adjunctive work beyond our scope and we focused on learning categorical/relational instances by using the contextual co-occurrence statistics within a static, predefined ontology. Nevertheless, they are all worth noting as they embody several future research directions.

Chapter 4

Rough Sets, Tolerance Approximation and TRSM

Rough Set theory was proposed by Zdzislaw Pawlak in 1980s [22, 23] as a new mathematical framework for reasoning about ill-defined objects. It is of fundamental importance to artificial intelligence (AI) and cognitive sciences, especially in the areas of machine learning, knowledge acquisition, decision analysis, knowledge discovery from databases, expert systems, decision support systems, inductive reasoning, and pattern recognition [24].

In this chapter, we restrict our discussion to classical rough sets (proposed by Z. Pawlak) and its tolerance rough set extension, which is the theory that forms the framework of this thesis. Subsequently, the tolerance rough sets model is introduced along with its applications in text and document clustering.

4.1 Rough Sets

In classical rough sets theory, a universe of objects is partitioned into indiscernible classes (i.e. granules) by means of an indiscernibility relation. Indiscernible classes form basic granules of knowledge about the universe. Given a concept that is determined to be vague (not precise), this theory makes it possible to express the vague concept by a pair of precise concepts called the lower and the upper approximation. A vague concept is defined as a class (or decision) that cannot be properly classified. The difference between the upper and the lower approximation constitutes the boundary region of the vague concept. Hence, rough set theory expresses vagueness

not by means of membership, but by employing a boundary region [24].

4.1.1 Formal Framework

Let U be a finite, non-empty universe of objects and let $R \subseteq U \times U$ denote a binary relation on the universe U. R is called an *indiscernibility relation* and for rough sets, it has to be an *equivalence relation*. The pair

$$(U, R) = \mathcal{A}$$

constitutes an *approximation space* [34]. Assume we have $X \subseteq U$ as a target concept in this universe. Then the task is to create an approximated representation for X in U with the help of R.

Let $[x]_R$ denote the indiscernibility class of x i.e. $y \in [x]_R \iff (x, y) \in R$. Then, every equivalence class forms a *granule* or *partition* which, as the name implies, contains objects that are indiscernible for this approximation space $\mathcal{A}$. Therefore, every single item in a granule is considered identical and inseparable. Eventually, these granules are approximated by the following means

- **Lower approximation**. Intuitively, these are the objects which certainly belong to X with respect to A.

$$\mathcal{L}_{\mathcal{A}}(X) = \{x \in U : [x]_R \subseteq X\} \tag{4.1}$$

- **Upper approximation**. Intuitively, these are the objects which may belong to X with respect to A.

$$\mathcal{U}_{\mathcal{A}}(X) = \{x \in U : [x]_R \cap X \neq \emptyset\} \tag{4.2}$$

These two approximations will also form the following two regions

- **Boundary region**. These are the objects occurring in the upper approximation but not in lower approximation of X.

$$\mathcal{B}_{\mathcal{A}}(X) = \mathcal{U}_{\mathcal{A}}(X) - \mathcal{L}_{\mathcal{A}}(X) \tag{4.3}$$

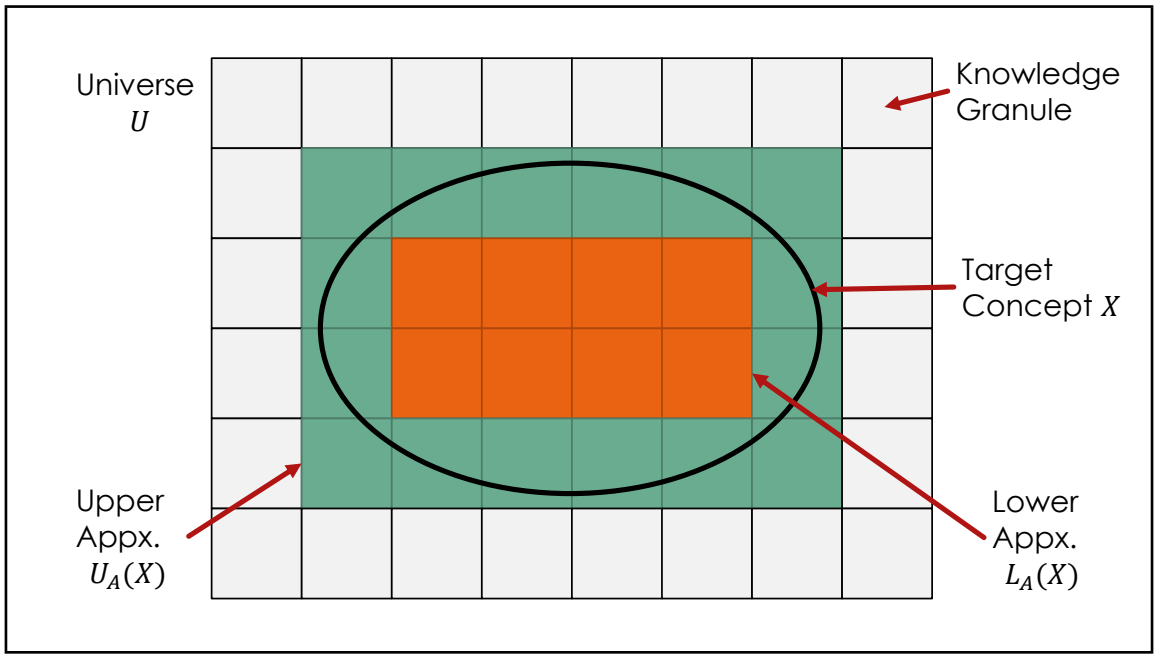


Figure 4.1: Rough sets and set approximation

- **Negative region**. These are the objects that certainly don't belong to X.

$$U - \mathcal{U}_{\mathcal{A}}(X) \tag{4.4}$$

Figure 4.1 shows the regions that emerge with set approximation. Orange granules constitute the lower approximation, granules in the green region make the boundary region, and orange and green granules combined form the upper approximation, leaving only the gray granules to form the negative region. We should note that each granule can contain an arbitrary number of objects or may be empty. They are depicted as squares only for the sake of illustration.

With this framework, we end up with two different types of sets: a set X is called a *crisp set* if and only if $\mathcal{B}_{\mathcal{A}}(X) = \emptyset$. Otherwise, it is called a *rough set*. The pair $(\mathcal{U}_{\mathcal{A}}(X), \mathcal{L}_{\mathcal{A}}(X))$ forms the *rough approximation* for X.

4.1.2 Rough Sets in Information Systems

So far, we have established the rough sets theory from a mathematical point of view. In this section, we illustrate its utility by means of a simple example.

Real world data is usually represented as an *information system* $I = (U, A)$ where U is a non-empty universe of *objects* and A is a non-empty finite set of *attributes*.

Every attribute $a \in A$ is associated to a function $f_a : U \to V_a$ where V_a is called the *value set* of a. Typically, an information system can be summarized within an information table where each row corresponds to an object (e.g. client, patient, subject,...) and each column corresponds to an attribute (e.g. age, income,...).

When there is a decision to be made for the objects in I, an additional attribute is appended to the information table and it becomes a *decision system*. It is characterized by the pair $(U, A \cup d)$ where $d \notin A$ is the *decision attribute*, and the elements of A are now called the *conditional attributes*. From the perspective of rough sets theory, the decision attribute is the target concept and the objects are represented as vectors of attribute values $v(x) = \langle f_{a_1}(x), f_{a_2}(x), ..., f_{a_n}(x) \rangle$ where $a_1, a_2, ..., a_n \in B \subseteq A$. The universe is then partitioned by an equivalence relation R over the set B

$$R = \{(x, y) \in U \times U : f_a(x) = f_a(y); \forall a \in B\} \tag{4.5}$$

By approximating the objects over the decision attribute, one can judge whether or not every object can be classified perfectly, by using the provided attributes in B.

Example Let $U = \{x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8\}$ be the universe of clients for a telecommunication company. Each client has an associated age and monthly income, as shown in the Table 4.1. In this case study, the decision attribute is whether or not they have subscribed to the cable tv service of that company. We can define an indiscernibility relation R over U in terms of the attributes 'Age' and 'Income': $R = \{(x, y) \in U \times U : f_{Age}(x) = f_{Age}(y) \wedge f_{Income}(x) = f_{Income}(y)\}$. It partitions U into the following five equivalence classes (granules):

$$\{x_1, x_8\}, \{x_2\}, \{x_3\}, \{x_4, x_5\}, \{x_6, x_7\} \tag{4.6}$$

Then, if we let $X = \{x \in U : Subscribed(x) = \text{'}yes\text{'}\}$, the upper and the lower approximations will be

$$\mathcal{U}_{\mathcal{A}}(X) = \{x_1, x_3, x_4, x_5, x_8\} \tag{4.7}$$

$$\mathcal{L}_{\mathcal{A}}(X) = \{x_3, x_4, x_5\} \tag{4.8}$$

This means that x_3, x_4 and x_5 can be certainly classified as "yes" and x_2, x_6 and x_7 can be certainly classified as "no".

Table 4.1: Sample decision system: Cable TV subscription

Client	**Age**	**Income**	**Subscribed**
x_1	30-40	\$2000-3000	yes
x_2	20-30	\$1000-2000	no
x_3	20-30	\$2000-3000	yes
x_4	40-50	\$2000-3000	yes
x_5	40-50	\$2000-3000	yes
x_6	20-30	\$500-1000	no
x_7	20-30	\$500-1000	no
x_8	30-40	\$2000-3000	no

4.2 Tolerance Approximation Spaces

As we have discussed, rough sets theory relies on an indiscernibility relation $R \subseteq U \times U$ to approximate a target concept and in the (classical) rough sets theory, R has to be an equivalence relation. In other words, it has the following 3 properties:

- Reflexivity: $(x, x) \in R \quad \forall x \in U$
- Symmetry: $(x, y) \in R \Rightarrow (y, x) \in R \quad \forall x, y \in U$
- Transitivity: $(x, y) \in R \wedge (y, z) \in R \Rightarrow (x, z) \in R \quad \forall x, y, z \in U$

In practice, it partitions the universe into disjoint (non-overlapping) equivalence classes which are regarded as information granules. However, there are some cases where the disjoint granules are not desired. Particularly, when it comes to natural language processing and information retrieval, we need a non-transitive binary relation that is reflexive and symmetric.

Example Consider the universe U of words {*account, agency, antecedent, backbone, backing, bottom, basis, cause, center, derivation, motive, root*} excerpted from Roget's thesaurus [11]. Assume we would like to define an indiscernibility relation R over those words based on their semantic affinity. Each of those words seem to share a meaning with one or more of the concepts *Root, Cause* and *Basis* and their meanings are not transitive. Therefore, overlapping classes would better fit to describe this universe and the desired outcome is shown in Figure 4.2.

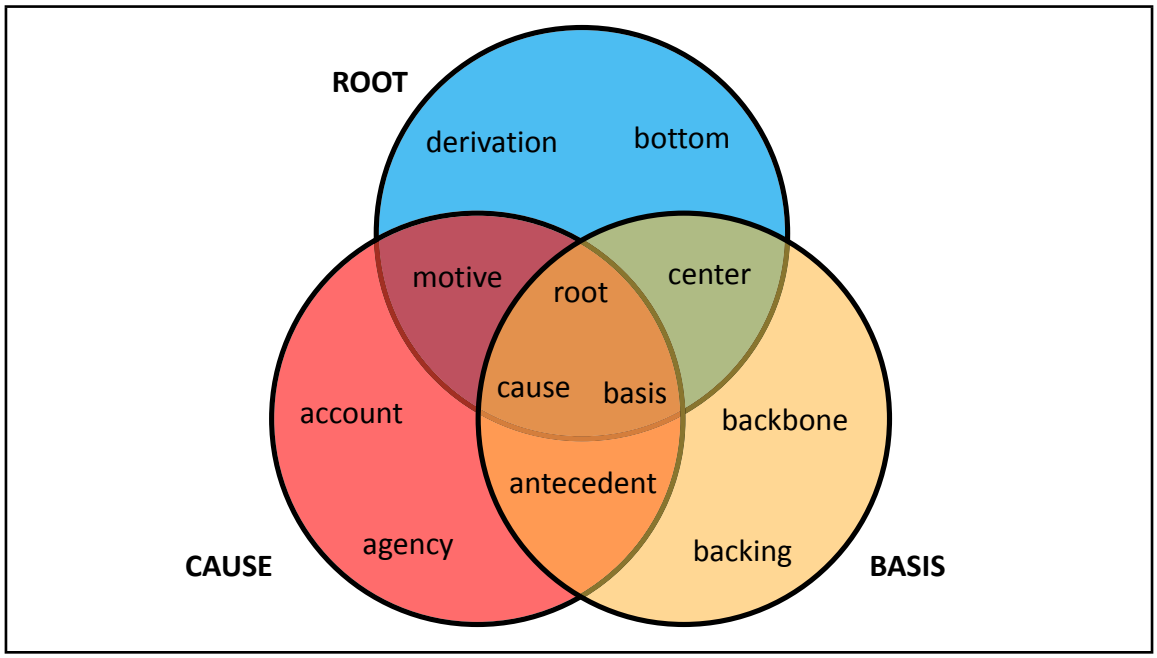


Figure 4.2: Overlapping classes of words [11]

In order to define overlapping classes or granules, we need *tolerance relations*. A tolerance relation $\mathcal{I} \subseteq U \times U$ can be any binary relation that is reflexive and symmetric. It will be used as an indiscernibility relation for a tolerance form of rough sets. Because it is not transitive, indiscernibility classes induced by such relations can overlap. For instance, the following non-transitivity is required for the classes in Figure 4.2: $(bottom, motive) \in \mathcal{I} \wedge (motive, account) \in \mathcal{I} \not\Rightarrow (bottom, account) \in \mathcal{I}$.

An indiscernibility class induced by a tolerance relation is called a *tolerance class*:

$$I(x) = \{y \in U \mid (x, y) \in \mathcal{I}\}$$

It is analogous to the equivalence class of the classical rough sets. There has been considerable effort in attempts to combine rough sets and tolerance relations ([29]) to obtain a realistic model (see for ex: [21, 27, 16]) leading to the tolerance rough sets model. In this work we use the tolerance approximation space model proposed in [34]. A tolerance approximation space [34] is denoted by

$$\mathcal{A} = (U, I, \nu, P) \tag{4.9}$$

where

- **Universe** U is the universe of objects.

- **Uncertainty function** $I : U \to \mathbb{P}(U)$ given that $\mathbb{P}(U)$ is the power set of U. It defines the *tolerance class* of an object. It also implicitly defines the tolerance relation $\mathcal{I}$ such that $x\mathcal{I}y \iff y \in I(x)$. It can be any relation that is reflexive and symmetric.

- **Vague inclusion function** $\nu : \mathbb{P}(U) \times \mathbb{P}(U) \to [0, 1]$ measures the degree of inclusion between two sets. It can be any function that is monotone with respect to the second argument: $Y \subseteq Z \implies \nu(X, Y) \leq \nu(X, Z)$ for $X, Y, Z \subseteq U$

- **Structurality function** $P : I(U) \to \{0, 1\}$ where $I(U) = \{I(x) : x \in U\}$ allows additional binary conditions to be defined over the tolerance classes.

Then, the lower and upper approximations of set X can be defined as:

$$\mathcal{L}_{\mathcal{A}}(X) = \{x \in U : P(I(X)) = 1 \wedge \nu(I(x), X) = 1\} \tag{4.10}$$

$$\mathcal{U}_{\mathcal{A}}(X) = \{x \in U : P(I(X)) = 1 \wedge \nu(I(x), X) > 0\} \tag{4.11}$$

What follows is a discussion of the Tolerance Rough Sets Model (TRSM) as a document representation model that is used for text clustering in general and for document clustering/classification in particular.

4.3 Document Representation with TRSM

Tolerance Rough Sets Model (TRSM) was proposed in [12, 11] for text clustering and document clustering/classification and to model relations between terms and documents [20]. Briefly, TRSM introduces a vectorial representation of documents where each vector dimension corresponds to a term weight that is to be enhanced by means of rough sets and tolerance approximation, by relating terms across documents. This is useful particularly when each document is characterized by only a small number of terms along with many zero-valued entries in a high dimensional term vector space. So TRSM promises a richer representation for documents to be clustered.

Consider a set of documents $D = \{d_1, d_2, ..., d_M\}$ and a universe of index terms $T = \{t_1, t_2, ..., t_N\}$ that occur in those documents. Each document d_j is to be represented

as a weight vector of its terms $\hat{d_j} = \langle (t_1, w_{1j}), (t_2, w_{2j}), ..., (t_N, w_{Nj}) \rangle$ where $w_{ij} \in [0, 1]$ shows the significance of term i in document j. Then, given a query Q as a cluster representative in the form $\hat{Q} = \langle (q_1, w_{1q}), (q_2, w_{2q}), ..., (q_s, w_{sq}) \rangle$ where $q_i \in T$ and $w_{iq} \in [0, 1]$, the task in hand is to find ordered documents $d_j \in D$ that are relevant to Q [11].

Determining the tolerance space. The tolerance approximation space $\mathcal{A} = (U, I, \nu, P)$ for documents is reconstituted as follows

- The universe is the set of index terms: $U = T = \{t_1, t_2, ..., t_N\}$
- The uncertainty function $I \subseteq T \times T$ aims to capture the affinity amongst the terms and defines the tolerance class for each index term. It is based on a tolerance relation that binds two terms if they co-occur frequently across documents. So the function becomes

$$I_\theta(t_i) = \{t_j | f_D(t_i, t_j) \geq \theta\} \cup \{t_i\} \tag{4.12}$$

  It is parametrized over a threshold value θ where $f_D(t_i, t_j)$ denotes the number of terms in which t_i and t_j co-occur. Note that $t_j \in I_\theta(t_i) \iff t_i \mathcal{I}_\theta t_j$ and that $\mathcal{I}_\theta$ is reflexive ($t_i \in I_\theta(t_i)$) and symmetric ($t_j \in I_\theta(t_i) \iff t_i \in I_\theta(t_j)$) for all $t_i, t_j \in T$, satisfying the tolerance relation requirements.

- The vague inclusion function is $\nu(X, Y) = \frac{|X \cap Y|}{|X|}$. It is monotonous w.r.t the second argument, as required. It can now be regarded as the membership function μ for term $t_i \in T$ to target concept $X \subseteq T$

$$\mu(t_i, X) = \nu(I_\theta(t_i), X) = \frac{|I_\theta(t_i) \cap X|}{|I_\theta(t_i)|} \tag{4.13}$$

- Provided that T is a closed set and Q consists exclusively of terms from T, the structurality function is simply $P = 1$ for TRSM.

The lower and upper approximations of X are defined as follows:

$$\mathcal{L}_\mathcal{A}(X) = \{t_i \in T : \frac{|I_\theta(t_i) \cap X|}{|I_\theta(t_i)|} = 1\} \tag{4.14}$$

$$\mathcal{U}_\mathcal{A}(X) = \{t_i \in T : \frac{|I_\theta(t_i) \cap X|}{|I_\theta(t_i)|} > 0\} \tag{4.15}$$

Weight Adjustment via Tolerance Rough Sets Tolerance approximation is used to enhance the document representation by adjusting the term weights. In the absence of such enhancement, term weights are assigned by using the term frequency-inverse document frequency (tf-idf) scheme

$$w_{ij} = \begin{cases} (1 + log(f_{d_j}(t_i))) \times log\frac{M}{f_D(t_i)} & \text{if } t_i \in d_j \\ 0 & \text{if } t_i \notin (d_j) \end{cases} \tag{4.16}$$

where $f_{d_j}(t_i)$ denotes the number of times t_i occurs in d_j (term frequency) and $f_D(t_i)$ denotes the number of documents in D that accommodates t_i (document frequency) [11]. In such a model, a term t_i acquires a nonzero weight for $\hat{d}_j$ if and only if it directly occurs in the document d_j. On the other hand, the upper approximation of a document $\mathcal{U}_{\mathcal{A}}(d_j)$ covers the "tolerant" terms for all of its own terms as well. So TRSM uses the following weighing scheme which also takes those boundary terms into account and assigns nonzero weights

$$w_{ij} = \begin{cases} (1 + log(f_{d_j}(t_i))) \times log\frac{M}{f_D(t_i)} & \text{if } t_i \in d_j \\ min_{t_h \in d_j} w_{hj} \times \frac{log(M/f_D(t_i))}{1+log(M/f_D(t_i))} & \text{if } t_i \in \mathcal{U}_{\mathcal{A}}(d_j) \backslash d_j \\ 0 & \text{if } t_i \notin \mathcal{U}_{\mathcal{A}}(d_j) \end{cases} \tag{4.17}$$

creating the enriched representation.

Clustering Documents Once the weights are adjusted within the framework of tolerance rough sets, one can measure the similarity between a query vector $\hat{Q}$ and a document vector $\hat{d}_j$ by using the following formula

$$Similarity(\hat{Q}, \hat{d}_j) = \frac{2 \times \sum_{k=1}^{N} (w_{kq} \times w_{kj})}{\sum_{k=1}^{N} w_{kq}^2 + \sum_{k=1}^{N} w_{kj}^2} \tag{4.18}$$

and ultimately, cluster the similar documents. A query vector may represent an actual query in the context of information retrieval or a class of documents in the context of document classification.

4.3.1 Various Approaches Using TRSM for Document Clustering

TRSM has been gaining popularity in document clustering and there are several methods which employ this model along with different approaches. In this section, we discuss the popular TRSM-based work for document clustering.

Hierarchical Document Clustering

The earliest known work on the application of TRSM as a document representation model was proposed by Kawasaki et al. They introduced a TRSM-based hierarchical document clustering that is an extension of the hierarchical agglomerative (bottom-up) clustering algorithm [12]. In this model, every document is represented as a weight vector of its terms and upper approximated by using a tolerance relation over the terms, as described by the TRSM framework in Eq. 4.17. As before, it aims to minimize the number of zero-valued coefficients in document vectors as well as to increase the degree of similarity between documents with few common terms. Once the representation is established, the clustering algorithm takes place. It first assigns each document to a different cluster and defines cluster representatives as supersets of popular terms of the constituting documents'. Subsequently, it finds the most similar pair of clusters (by using a similarity method such as Dice, Jaccard or Cosine) and merges them, in an iterative fashion, until all the clusters are merged into an ultimate single cluster. The advantage of using a hierarchy is that it allows the use of document cluster representatives to calculate the similarity between clusters instead of averaging similarities of all document pairs included in clusters, which aids the execution time [12]. The results of validation and evaluation of this method suggest that this clustering algorithm can be well adapted to text mining [12].

Non-hierarchical Document Clustering

Soon after, Ho et al. introduced a non-hierarchical document clustering method using TRSM [11]. The authors pointed out that hierarchical methods become unsuitable for large document corpora, due to exploding time and space requirements of the underlying algorithms [11]. This model also uses the TRSM framework described in Section 4.3 and forms a pre-specified number of possibly overlapping document clusters. First, the TRSM-based document representation is established (documents

are approximated using the upper approximation operator, term weights are adjusted according to Eq. 4.17). Then, the cluster representatives R_k are formed by randomly assigning a document to each cluster. Similarly to the hierarchical approach, this is done by using the popular terms of the constituting documents. Next, the similarity between each cluster rep. and the upper approximation of each document is calculated, as shown in Eq. 4.18. If the similarity is above a given threshold, the document is also assigned to that cluster, and the cluster rep. is recalculated. This process continues until there is no more change in the clusters. The algorithm has been evaluated and validated by experiments on test collections [11].

Lexicon-based Document Clustering

More recently, a novel method for document clustering, named a lexicon-based document representation (LBDR) was presented by Virginia et al. [40]. This model uses TRSM in presence of a lexicon with the intention of creating an enhanced but also a compact document representation. First of all, LBDR creates a term weight vector for each document and then enhances the representation by means of TRSM, just like the hierarchical [12] and non-hierarchical [11] methods. Next, the terms are mapped to a lexicon and the ones which don't occur in the lexicon (i.e. irrelevant, non-informative terms) are filtered out reducing the number of dimensions in the vectors, creating the compact but yet enhanced representation. The intuition behind this approach can be demonstrated via Figure 4.3. In Figure 4.3 (a), we can see how document d_1 and the lexicon overlap. The intersection is compact, but limited. In Figure 4.3 (b), the dashed line shows the upper approximated TRSM representation of d_1. LBDR combines the two and creates the dense and enhanced representation of d_1 in lower dimensional space, as shown in the dark shaded area in Figure 4.3 (c). Eventually, the authors conclude that the effectiveness of lexicon-representation is comparable with TRSM-representation while the efficiency of lexicon-representation should be better than the existing TRSM-representation [40].

Web Search Results Clustering

Ngo and Nguyen [20] focused on a more specific type of document clustering. They proposed a web search results clustering method which is based on tolerance rough sets model. Their goals were the same as in [11] and [12], creating an enriched representation for the web documents in order to reveal the subtle inter-document

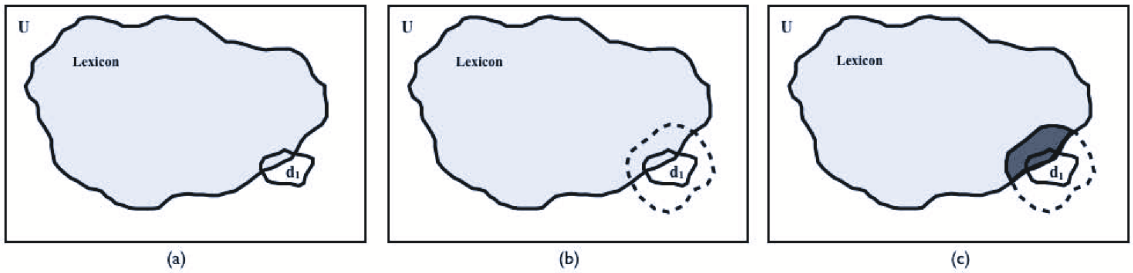


Figure 4.3: Lexicon-based document representation [40]

similarities and to boost the clustering quality. They proposed a Tolerance Rough set Clustering (TRC) algorithm, which is based on k-means clustering. First, each document is pre-processed to create an index term-based vectorial representation. After that, those vectors are combined and a term-document matrix is formed. Then, they enhance the term weights of the documents by using TRSM and upper approximation. Ultimately, TRC clusters the search results and labels them on a given query. Their experiments have shown that tolerance rough sets and upper approximation it offers can indeed improve the representations, with positive effects on the clustering quality [20].

Two Class Clustering with Ensemble Learning

Shi et al. [33] proposed a tolerance-based semi-supervised two-class ensemble classifier for documents when there are positive and unlabelled examples present, yet no negatives. They used tolerance rough sets to extract a set of negative examples upon which they built an ensemble classifier using Naive Bayes and Support Vector Machine algorithms. Experimental results indicate that the proposed method achieves significant performance improvement [33].

Chapter 5

A Semi-supervised Tolerance Rough Sets Approach for Web Information Labeling

In this chapter, we present the formal model and the algorithm for our proposed tolerant pattern learner (TPL) [31, 32], whose task is to address the Nell problem described in Section 3.2, that is, to extract

i) categorical noun phrase instances (e.g. *City(Winnipeg)*)

ii) relational noun phrase pairs (e.g. *CapitalOf(Canada,Ottawa)*)

using contextual co-occurrence statistics between the noun phrases (e.g. *Winnipeg*) or noun phrase pairs (e.g. *(Canada,Ottawa)*) and the contextual patterns (e.g. *"North American cities such as arg"*, *"arg1 is the capital of arg2"*), parsed from a web derived data set. These two tasks are handled separately by the categorical and relational extractor modules of TPL, respectively.

TPL is our attempt to employ the tolerance rough sets model (TRSM) to the semi-supervised web information labeling problem. As we have discussed in Section 4.3 TRSM has been successfully used as a document representation tool to serve the task of clustering related documents. In all the work discussed in that section, documents were the target entities and the index terms were the features describing the documents. In this research, instead of documents, the target entities are the noun phrases and instead of index terms, the features are the contextual patterns. In other words, we observed that there is a natural affinity between the document clustering problem and the context-based noun phrase clustering problem.

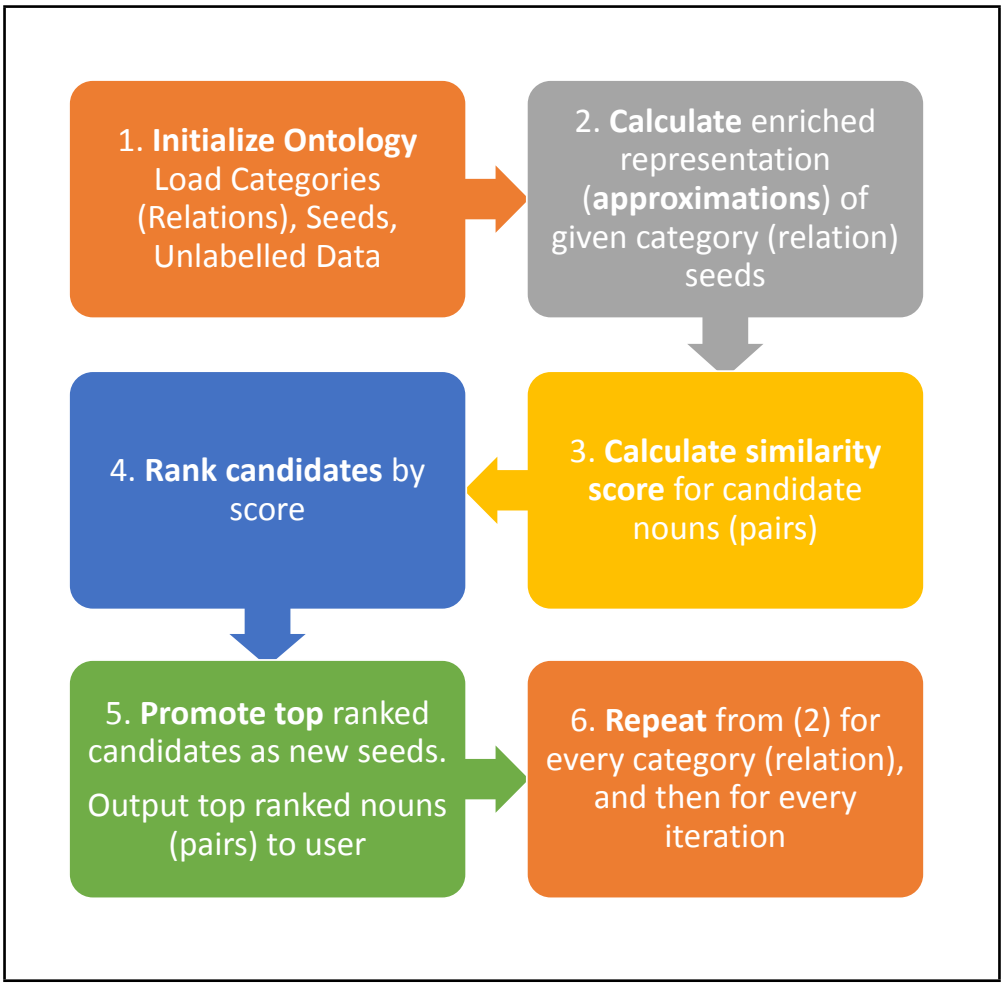


Figure 5.1: TPL algorithm flow

TPL shown in Figure 5.1 represents noun phrases in terms of their co-occurring contextual patterns, and it enhances this representation by using the two approximation operators, upper and lower approximations on noun phrases via a tolerance relation defined over those contexts. Ultimately, it detects the most likely candidates for a given category by scoring them in terms of their similarity with the trusted instances. The top instances are promoted and fed back to the system as new seeds, completing the semi-supervised framework. The relational fact extraction is analogous to this categorical extraction process. In that case, instances are semantically related noun phrase pairs and the predicates are relations instead of categories.

Rest of this chapter describes the formal framework and the algorithms of the TPL model, for both categorical and relational fact extractors.

5.1 Formal framework

Let us first define the universes of entities to operate on:

- $\mathcal{N} = \{n_1, n_2, .., n_M\}$ is the universe of noun phrases. This set will accommodate every single noun phrase to be parsed from the source web documents.
- $\mathcal{C} = \{c_1, c_2, ..., c_P\}$ is the universe of categorical (unary) contextual patterns. These contexts are to yield the individual noun phrases to be extracted as category instances.
- $\mathcal{R} = \{r_1, r_2, ..., r_Q\}$ is the universe of relational (binary) contextual patterns. These contexts are to yield the noun phrase pairs to be extracted for relations.
- $\mathcal{T} = \{t_{ij} = (n_i, n_j) \in \mathcal{N}^2 : \exists r_k \in \mathcal{R} \wedge f_{\mathcal{T}}(t_{ij}, r_k) > 0\}$ is the universe of co-occurring noun phrase pairs (i.e. tuples) described via the relational co-occurrence function $f_{\mathcal{T}}(t_{ij}, r_k) = \{\kappa \in \mathbb{N} : t_{ij}$ occurs κ times within the context $r_k\}$

We can define the following cross-mapping functions to represent every noun phrase (and noun phrase pair) by means of their contexts, and vice versa:

- $C : \mathcal{N} \rightarrow \mathbb{P}(\mathcal{C})$ maps each noun phrase to its set of co-occurring categorical contexts: $C(n_i) = \{c_j : f_{\mathcal{N}}(n_i, c_j) > 0\}$ where $f_{\mathcal{N}}(n_i, c_j) = \{\kappa \in \mathbb{N} : n_i$ occurs κ times within context $c_j\}$
- $N : \mathcal{C} \rightarrow \mathbb{P}(\mathcal{N})$ maps each categorical context to its set of co-occurring noun phrases: $N(c_j) = \{n_i : f_{\mathcal{N}}(n_i, c_j) > 0\}$
- $R : \mathcal{T} \rightarrow \mathbb{P}(\mathcal{R})$ maps each noun phrase pair to its set of co-occurring relational contexts: $R(t_{ij}) = \{r_k : f_{\mathcal{T}}(t_{ij}, r_k) > 0\}$
- $T : \mathcal{R} \rightarrow \mathbb{P}(\mathcal{T})$ maps each relational context to its set of co-occurring noun phrase pairs: $T(r_k) = \{t_{ij} : f_{\mathcal{T}}(t_{ij}, r_k) > 0\}$

With the help of these cross-mapping functions, we can define the following approximation spaces to provide the framework for the categorical and relational information extraction, respectively.

Definition 1. A *categorical noun-context tolerance model* [31] is described by the tolerance approximation space $\mathcal{A} = (\mathcal{C}, \mathcal{N}, I, \omega, \nu)$ where $\mathcal{N}$ and $\mathcal{C}$ are as defined

previously. $I = I_\theta$ is the parametrized uncertainty function describing the tolerance classes for the contexts, in terms of contextual overlaps:

$$I_\theta(c_i) = \{c_j : \omega(N(c_i), N(c_j)) \geq \theta\} \tag{5.1}$$

Here, θ is the tolerance threshold. ω is the overlap index which is the Sorensen-Dice index [35]:

$$\omega(A, B) = \frac{2|A \cap B|}{|A| + |B|} \tag{5.2}$$

$\nu : \mathbb{P}(\mathcal{C}) \times \mathbb{P}(\mathcal{C}) \to [0, 1]$ measures the degree of inclusion and is defined as $\nu(X, Y) = \frac{|X \cap Y|}{|X|}$. Within the framework of $\mathcal{A}$, a context-described noun phrase can now be approximated using the lower approximation:

$$\mathcal{L}_\mathcal{A}(n_i) = \{c_j \in \mathcal{C} : \nu(I_\theta(c_j), C(n_i)) = 1\} \tag{5.3}$$

giving us its "closely" related contexts; or else it can be approximated with the upper approximation to its "somewhat" related contexts:

$$\mathcal{U}_\mathcal{A}(n_i) = \{c_j \in \mathcal{C} : \nu(I_\theta(c_j), C(n_i)) > 0\} \tag{5.4}$$

Definition 2. A *relational noun-context tolerance model* [32] is the analogous model to extract related pairs. It is described by the approximation space $\mathcal{A} = (\mathcal{R}, \mathcal{T}, I, \omega, \nu)$ where $\mathcal{T}$, $\mathcal{R}$, ω and ν are defined as previously. I_θ is again the uncertainty function with the tolerance threshold θ:

$$I_\theta(r_i) = \{r_j : \omega(T(r_i), T(r_j)) \geq \theta\} \tag{5.5}$$

Within the framework of $\mathcal{A}$, a context-described noun phrase pair can now be lower approximated to its closely related contexts:

$$\mathcal{L}_\mathcal{A}(t_{ij}) = \{r_k \in \mathcal{R} : \nu(I_\theta(r_k), R(t_{ij})) = 1\} \tag{5.6}$$

or else it can be upper approximated to its somewhat related contexts:

$$\mathcal{U}_\mathcal{A}(t_{ij}) = \{r_k \in \mathcal{R} : \nu(I_\theta(r_k), R(t_{ij})) > 0\} \tag{5.7}$$

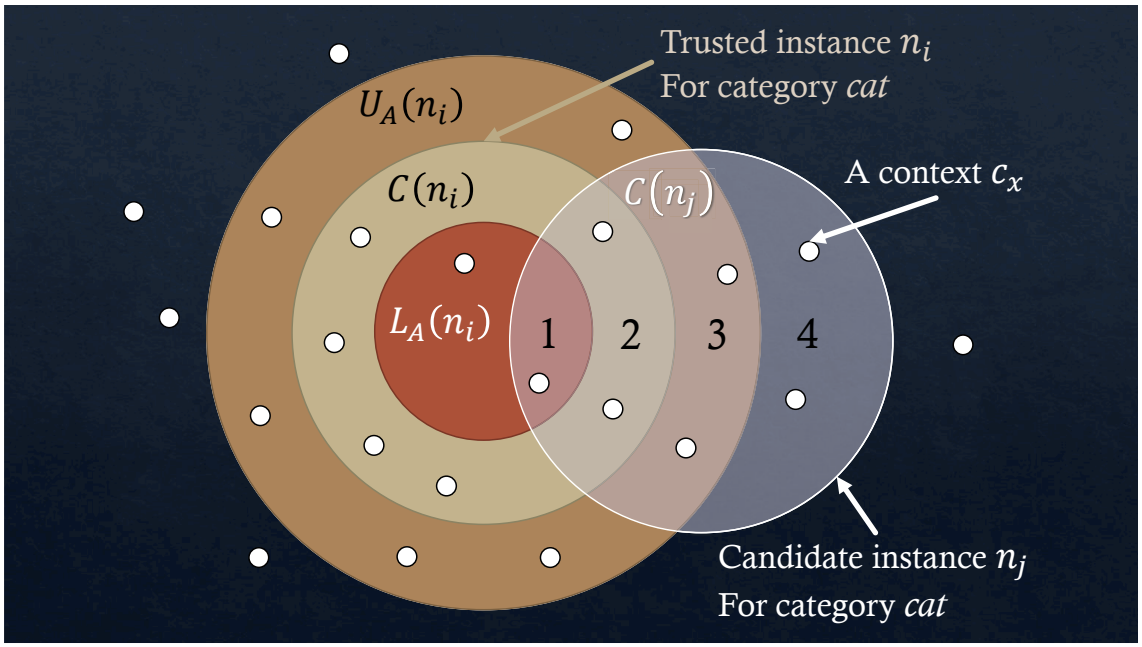


Figure 5.2: Four zones of recognition for contexts emerging from approximations of n_i.

A tolerance approximation space is employed by TRSM to generate an enriched representation for the documents. In an analogous way, it is used by TPL to create an enriched representation for the promoted noun phrases and pairs. Nevertheless, TPL does not use a vector-space model and it describes noun phrases as sets of co-occurring contexts, instead of vectors. In accordance, we associate every trusted instance n_i of a given category *cat* to these following three descriptor sets: $C(n_i)$, $\mathcal{U}_\mathcal{A}(n_i)$ and $\mathcal{L}_\mathcal{A}(n_i)$. These sets are employed to calculate a *micro-score* for the candidate noun phrase n_j, against the trusted instance n_i of the category *cat*:

$$\begin{aligned} micro(n_i, n_j) = \omega(C(n_i), C(n_j))\alpha + \\ \omega(\mathcal{U}_\mathcal{A}(n_i), C(n_j))\beta + \omega(\mathcal{L}_\mathcal{A}(n_i), C(n_j))\gamma \end{aligned} \tag{5.8}$$

Once more, we make use of the overlap index function ω in Eq. 5.2 for this calculation. α , β and γ are the contribution factors of the scoring components and they may be adjusted for the particular application domain.

The intuition behind this approach may be explained by Figure 5.2. A trusted instance n_i has the universe of contexts partitioned by its descriptors $\mathcal{L}_\mathcal{A}(n_i)$, $C(n_i)$

and $\mathcal{U}_{\mathcal{A}}(n_i)$ into four zones of recognition. For a candidate n_j, each zone will represent a different degree of similarity. When calculating the micro-score, the candidate's contexts falling in zone 1 (lower approximation) will be covered by all three descriptors and will thus make a high contribution to its score. Contexts in zone 2 will be covered by $C(n_i)$ and $\mathcal{U}_{\mathcal{A}}(n_i)$ so they will make medium contribution. Zone 3 contexts will only be covered by $\mathcal{U}_{\mathcal{A}}(n_i)$ and they will make low contribution. Contexts in zone 4 will not to contribute at all since they suggest no resemblance between n_i and n_j.

An analogous scoring mechanism is also employed for learning relations. These descriptors are used to calculate a micro-score for a candidate pair t_{kl}, by the trusted pair t_{ij}:

$$\begin{aligned} micro(t_{ij}, t_{kl}) = \omega(C(t_{ij}), C(t_{kl}))\alpha + \\ \omega(\mathcal{U}_{\mathcal{A}}(t_{ij}), C(t_{kl}))\beta + \omega(\mathcal{L}_{\mathcal{A}}(t_{ij}), C(t_{kl}))\gamma \end{aligned} \tag{5.9}$$

5.1.1 Categorical Noun Phrase Extractor Algorithm

The input for the categorical extractor is an ontology which is formed by a set of categories (e.g. *City*) and a handful of seed noun phrases (e.g. *Winnipeg, New Delhi, Ankara*). Furthermore, it expects a large co-occurrence matrix representing the noun phrases and the contextual patterns extracted from the world wide web. The output are trusted instances assigned to their respective categories within the ontology.

In this design, a category is represented by means of its trusted instances. Hence, a trusted instance acts as a "proxy" for the category it belongs. Candidate noun phrases are ordered by their similarity to these proxies, and thus, to the categories. Seed instances are to serve as the initial proxies.

TPL employs a score-based ranking and the scoring mechanism is given in Eq. 5.8. For a given category *cat*, we can keep a *macro-score* (*i.e.* an accumulated micro-score of proxies) for the candidate n_j

$$macro_{cat}(n_j) = \sum_{\forall n_i \in Trusted_{cat}}^{n} micro(n_i, n_j) \tag{5.10}$$

After calculating it for every candidate of *cat*, we rank the candidates by their macro-scores (normalized by the number of trusted instances of *cat*). Eventually, we promote the top new candidates as trusted.

The overall flow for the categorical extractor is summarized in Algorithm 3. Like

CBS and CPL, TPL is an iterative algorithm which is appointed to run indefinitely. In every iteration r, it learns new trusted instances of categories and it uses its growing knowledge to make more informed judgments. As the first iteration is based on user-labelled seeds, it forms the supervised step of the algorithm. The succeeding iterations use the automatically extracted instances, forming the unsupervised step.

Algorithm 3: Tolerant Pattern Learner for Categories

Input : An ontology O defining categories and a small set of seed examples; a large corpus U

Output: Trusted instances for each category

```
for r = 1 → ∞ do
    for each category cat do
        for each new trusted noun phrase n_i of cat do
            Calculate the approximations U_A(n_i) and L_A(n_i);
            for each candidate noun phrase n_j do
                Calculate micro(n_i, n_j);
        for each candidate noun phrase n_j do
            macro_cat(n_j) = Σ_{∀n_i ∈ cat} micro(n_i, n_j) (Eq. 5.10);
        Rank instances by macro_cat/|cat|;
        Promote top instances as trusted;
```

5.1.2 Relational Noun Phrase Pair Extractor Algorithm

Similarly, the input for the relational extractor is an ontology formed by a set of relations (e.g. *City-Country*) as well as a few seed noun phrase pairs per relation (e.g. *(Winnipeg, Canada), (New Delhi, India), (Ankara, Turkey)*). It also expects a large co-occurrence matrix representing the noun phrase pairs, and the contextual patterns. The output are trusted relation instances, in forms of ordered noun phrase pairs, assigned to their respective relations.

An analogous score-based ranking is also employed for relations. For the relation rel, an accumulated macro-score is maintained for the candidate t_{kl}

$$macro_{rel}(t_{kl}) = \sum_{\forall t_{ij} \in Trusted_{rel}}^{n} micro(t_{ij}, t_{kl}) \tag{5.11}$$

The overall flow for the relational noun phrase extractor is summarized in Algo-

rithm 4. It functions with the same principles as the categorical extractor. Each trusted pair represents its associated relation and is used to detect and extract more pairs in an iterative manner.

Algorithm 4: Tolerant Pattern Learner for Relations

Input : An ontology O defining relations and a small set of seed examples; a large corpus U

Output: Trusted pairs for each relation

for $r = 1 \rightarrow \infty$ **do**
  for each relation rel **do**
    for each new trusted noun phrase pair t_{ij} of rel **do**
      Calculate the approximations $\mathcal{U}_{\mathcal{A}}(t_{ij})$ and $\mathcal{L}_{\mathcal{A}}(t_{ij})$;
      for each candidate noun phrase pair t_{kl} **do**
        Calculate $micro(t_{ij}, t_{kl})$;
    for each candidate noun phrase pair t_{kl} **do**
      $macro_{rel}(t_{kl}) = \sum_{\forall t_{ij} \in rel} micro(t_{ij}, t_{kl})$ (Eq. 5.11);
    Rank pairs by $macro_{rel}/|rel|$;
    Promote top pairs as trusted;

Chapter 6

Implementation, Experiments, Results and Discussion

This chapter discusses the implementation details, experiments and empirical results for our proposed Tolerant Pattern Learner, for both categorical and relational fact extractions. We also provide further discussion on our findings regarding TPL's performance, compared to the benchmark algorithms CBS [39] and CPL [3].

6.1 Category Instance Extraction

6.1.1 Implementation

The categorical noun phrase extractor module of TPL was implemented in MATLAB®. We chose this environment because the dataset TPL expects is in form of a sparse (noun phrase-contextual pattern co-occurrence) matrix, which is the primary type of data MATLAB® was designed to operate on. The experiments for the benchmark algorithm CBS were also conducted in MATLAB® [39] so it seemed to be a proper choice for this task. For the experiments, we used a Windows 7 machine with 2.40 GHz Intel i7 Processor with 16 GB of memory (4 GB available).

6.1.2 Dataset

The original source for our data matrix is ClueWeb09 [2]. It is a massive web document collection consisting of roughly 1 billion web pages in ten languages, collected in early 2009 [2]. Nonetheless, we did not directly interact with these web pages. We

Table 6.1: All-pairs data set summary

File	Information	# Elements	Size on Disk
F_1	Noun phrase list	≈ 9.8 Million	≈ 185 MB
F_2	Contextual pattern list (categorical)	≈ 8.5 Million	≈ 193 MB
F_3	Noun-Context co-occurrence counts	≈ 1.1 Billion	≈ 26.4 GB
F_4	Noun phrase pair list	≈ 144 Million	≈ 3.6 GB
F_5	Contextual pattern list (relational)	≈ 11 Million	≈ 350 MB
F_6	Pair-Context co-occurrence counts	≈ 1.8 Billion	≈ 46.9 GB

Table 6.2: All-pairs data set format

File	Data Format
F_1	$\langle n_i \rangle$ \t $\langle$count$\rangle$ \n
F_2	$\langle c_i \rangle$ \t $\langle$count$\rangle$ \n
F_3	$\langle n_i \rangle$ [\t $\langle c_j \rangle$ -#- $\langle$count$\rangle$ [...]] \n
F_4	$\langle n_i \rangle \| \| \langle n_j \rangle$ \t $\langle$count$\rangle$ \n
F_5	$\langle r_i \rangle$ \t $\langle$count$\rangle$ \n
F_6	$\langle n_i \rangle \| \| \langle n_j \rangle$ [\t $\langle r_k \rangle$ -#- $\langle$count$\rangle$ [...]] \n

used the all-pairs data set [5] by Andy Carlson who is a member of the Nell team. Natural language processing methods were used to process pages from CluWeb09 to derive the all-pairs set which accommodates millions of raw noun phrases, contextual patterns and all the co-occurrence information. Table 6.1 summarizes the content of the all-pairs corpus. The information is organized within 6 different text files. Literals for noun phrases, noun phrase pairs and contextual patterns are stored in line-delimited lists along with their hit counts whereas the co-occurrence information is stored in a slightly more complicated format (see Table 6.2). Figure 6.1 is a snapshot for the noun phrase-contextual pattern co-occurrence information excerpted from file F_3 of the all-pairs data set. In every line, first entry is a noun phrase followed by a contextual pattern and the co-occurrence cardinality in between.

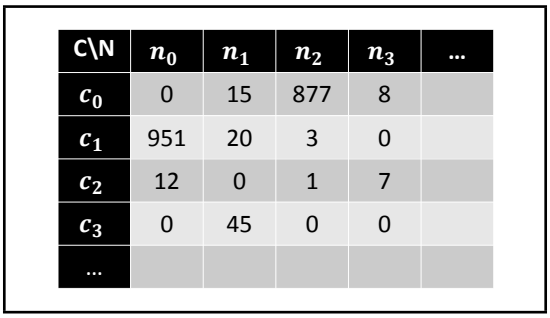
Data Preparation Procedure

Having access to the All-pairs data set was a big help for our experiments. However, the data set had to undergo significant pre-processing to be useful for TPL algorithm. We had to convert the data to a co-occurrence matrix which MATLAB® can read.

Figure 6.1: Noun-context co-occurrence counts (indexed by noun phrases) excerpted from all-pairs data set. Arrows denote delimiting tabs.

Moreover, we had to subsample the data as it was too large to be processed on the hardware available to our experiments. Fortunately, S.Verma provided us the 68,919 noun phrases used in the CBS experiments [39] so we decided to use them as our noun phrase universe $\mathcal{N}$. Rest of the information was to be acquired from the all-pairs dataset and we took the following approach:

1. First, we exported those 68,919 noun phrases from MATLAB® to a list in text form.
2. Next, we coded a Java applet to extract, for each of those noun phrases, the list of the top 100 co-occurring contexts from the all-pairs set, in textual form.
3. Subsequently, we went over those co-occurring context lists, eliminated duplicates and populated our initial context universe $\mathcal{C}$ accommodating 930,426 contexts, in textual form.
4. Next, we imported $\mathcal{N}$ and $\mathcal{C}$ back to MATLAB® as 1-D arrays of string literals.
5. Then, we initialized an empty *noun* $\times$ *context* co-occurrence matrix with dimensions $|\mathcal{C}| \times |\mathcal{N}|$.
6. We went over the co-occurrence lists once more and imported the co-occurrence



C\N	n_0	n_1	n_2	n_3	...
c_0	0	15	877	8	
c_1	951	20	3	0	
c_2	12	0	1	7	
c_3	0	45	0	0	
...					

Figure 6.2: Noun-context co-occurrence matrix

cardinalities to our matrix.

7. For further feasibility, we eliminated the contexts with low cross-noun phrase frequency value i.e. $|N(c_i)| < 10$.

It took us roughly 12 hours to execute all these steps, accumulatively. At the end, we obtained a dataset consisting of 68,919 unique noun phrase instances and 59,325 unique contextual patterns stored in a matrix M_{ij} with each cell recording the co-occurrence cardinality of contextual pattern i against noun phrase j, as shown in Figure 6.2.

6.1.3 Experimental Configuration

Throughout our experiments for the category extractor, we attempted to pursue the same conventions as CBS [39]. We used the same 11 categories in our ontology: *Company, Disease, KitchenItem, Person, PhysicsTerm, Plant, Profession, Sociopolitics, Website, Vegetable, Sport.* Each category was initialized by 5-8 seed instances and we let the extractor run for 10 iterations. For every category, we had the top 5 new noun phrases promoted as "trusted" per iteration, which were then destined to be used as seeds in the upcoming iterations. We heuristically set the tolerance threshold to $\theta = 50\%$ since it led us to the most semantically accurate tolerance classes. We also tried to provide a balance between directly co-occurring terms and their tolerants by setting the contribution factors to $\alpha = 0.5, \beta = 0.25$ and $\gamma = 0.25$.

6.1.4 Evaluation Criterion

The metric we used to measure the performance of our category extractor is *Precision@30*, which is also the same metric used to evaluate CBS [39]: In any iteration,

Table 6.3: Precision@30 of TPL for all categories, by iteration (%)

Iteration	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Company	100	100	100	100	100	100	100	100	100	100
Disease	100	100	100	100	100	100	100	100	100	100
KitchenItem	100	100	100	100	100	100	100	100	100	100
Person	100	100	100	100	100	100	100	100	100	100
PhysicsTerm	100	100	100	100	93	97	97	93	90	90
Plant	100	100	97	97	100	97	97	97	97	97
Profession	100	100	100	97	100	100	100	100	100	100
Sociopolitics	100	100	100	100	100	100	100	100	100	100
Website	87	90	90	90	90	90	90	90	90	90
Vegetable	77	90	90	90	93	93	87	80	73	63
Sport	100	97	97	97	97	97	97	100	100	100
Average	**96.7**	**97.9**	**97.6**	**97.3**	**97.5**	**97.6**	**97.3**	**96.4**	**95.4**	**94.5**

after scoring and ranking noun phrases for a given category, we compute the percentage of the correct instances in the set of top-30 ranked noun phrases. At the end, we manually judged the accuracy of the extractions, since the data was unlabelled.

6.1.5 Results

We ran our test scenario as a single thread, on a Windows 7 machine with 2.40 GHz Intel i7 processor; it took around 2 hours and 40 minutes. Table 6.3 summarizes the Precision@30 results for every category per iteration and the outcome is promising. For vast majority of the categories, TPL successfully maintained high precision over 10 iterations without showing any sign of semantic drifting. One category that yielded a relatively poor precision was *Vegetable*; the algorithm first drifted by mistaking fruits for vegetables and then went considerably off-track by labeling meat products and dairies. (See Table 6.4.) However, it was also a challenging category for the CBS algorithm so this problem may be deemed as a categorical anomaly.

TPL managed to achieve a comparable performance with CBS, in terms of the precision metric. Table 6.5 illustrates the results for iterations 5 and 10. As the experimental setups (seeds, dataset) were not identical, a numerical comparison with CBS may not be very demonstrative. Nevertheless, the stable average precision values across iterations shall be considered as an indication for the promising nature of this granular-based algorithm.

Table 6.4: TPL's top-16 ranked instances for selected categories. Incorrect instances are boldfaced.

Iteration 1			Iteration 10		
Phys'Terms	**Soc'politics**	**Vegetables**	**Phys'Terms**	**Soc'politics**	**Vegetables**
inertia	socialism	zucchini	density	humanism	zucchini
acceleration	democracy	spinach	conductivity	pluralism	cabbage
gravity	dictatorship	cucumber	intensity	federalism	kale
buoyancy	monarchy	tomato	viscosity	interna'lism	celery
velocity	independence	broccoli	permeability	nationalism	cauliflower
momentum	justice	lettuce	velocity	rationality	eggplant
magnetism	equality	celery	brightness	liberalism	carrots
resonance	pluralism	cabbage	attenuation	secularism	asparagus
curvature	interna'lism	kale	luminosity	individualism	tomatoes
electromagnet.	federalism	cauliflower	reflectance	democracy	spinach
density	secularism	asparagus	sensitivity	environ'ism	squash
elasticity	liberalism	carrots	amplitude	morality	cucumber
surface tension	hegemony	tomatoes	thickness	pragmatism	**melon**
polarization	self-determ.	**avocado**	frequency	spirituality	**chicken**
vibration	unification	eggplant	**water cont.**	regionalism	**tofu**
entropy	capitalism	carrot	**salinity**	subjectivity	**shrimp**

6.1.6 Discussion

There seems to be a number of factors playing a role in the performance of TPL over individual categories. To begin with, some categories are more difficult to define and their boundaries are relatively vague, so the measured performance may alter depending on the inclusion degree. To illustrate, *Sociopolitics* was such a category that had no apparent borderline and for that category, judges accepted virtually every concept having an interpretation in a socio-political context (See Table 6.4). On the other hand, several nouns were rejected for the category *Website* even though they might have referred to websites but their sensible meanings were different (e.g. *Skype*^{TM}, *Microsoft*^{TM}). Propitiously, most categories including *Person*, *Sport*, *Disease*, *Plant*, *Company*, *Profession* had clear definitions and conceivable boundaries so the verdicts for their instances were much more apparent. Another factor that affects the performance of our category learner is the ratio of the inter-intra category similarity. *PhysicsTerm* is a good example for this case; the mis-extractions were chemistry terms (e.g. *reactivity*), or mathematical terms (e.g. *curvature*) which were relatively close to the context of physics. We already explained, some vegetables ap-

Table 6.5: Precision@30 of TPL and CBS per category. CBS results are as seen in [39]

Categories	**Iteration 5**		**Iteration 10**	
	TPL	**CBS [39]**	**TPL**	**CBS [39]**
Company	100%	100%	100%	100%
Disease	100%	100%	100%	100%
KitchenItem	100%	94%	100%	94%
Person	100%	100%	100%	100%
PhysicsTerm	93%	100%	90%	100%
Plant	100%	100%	97%	100%
Profession	100%	100%	100%	87%
Sociopolitics	100%	48%	100%	34%
Sport	97%	97%	100%	100%
Website	90%	94%	90%	90%
Vegetable	93%	83%	63%	48%
Average	**97.5%**	**92%**	**94.5%**	**87%**

peared to overlap with fruits, meat products and dairy because in essence, the main cluster those nouns tend to form is "food" and it challenged the algorithm.

As Verma et al. [39] point out, it is an arduous task to learn classifiers individually. For this reason, CBS and CPL coupled the learning procedure by simultaneously learning the classifiers. In particular, they enforced mutual exclusion constraints: given a pair of mutually exclusive categories A and B, evidence for a noun phrase to fall in the borders of A is used to decrease its likelihood to be assigned to B. Such a constraint has not been used for our learner so it may be considered as a potential room for improvement.

6.2 Relation Pair Extraction

6.2.1 Implementation

Using the libraries and the matrix infrastructure of MATLAB® came quite handy both for preparing the data and implementing the categorical noun phrase extractor. Accordingly, our first intention was to extend our category learner in MATLAB® to implement the relation learner module. Due to the nature of noun phrase pairs, our data matrix had to be much larger and MATLAB® emerged to be unsuitable for this task. So we implemented the relation-pair extractor in C++ as a 64-bit Windows application. For the implementation and the experiments, we used a Windows 7

Table 6.6: Arrays used to represent an efficiently traversable sparse matrix

data[]	Array of nonzero co-ocurrence values (row-major order)
rows[]	The row # of the cells in data array
cols[]	The column # of the cells in data array
nextCol[]	Index for the next element in the same column
rowHeader[]	Index for the header cell of every row
colHeader[]	Index for the header cell of every column

machine with 2.40 GHz Intel i7 processor and 16GB memory.

Sparse Representation Designed for the Data Matrix

This platform change introduced some additional design concerns. Representing the data matrix was one of them. MATLAB® used its own internal structures to efficiently store and access matrices in sparse forms so this part of the problem was abstracted from our work. With C++, however, we had to take care of this task ourselves.

We designed a sparse matrix framework in which the nonzero matrix cells are stored in a long 1-D array called *data*, along with the information in Table 6.6. This representation allows efficient row-by-row and column-by-column traversal of nonzero cell values, which are needed to find the co-occurring contexts of a noun phrase, the tolerance class of a context and the lower/upper approximations of a noun phrase.

6.2.2 Dataset

Similar to our experiments with category instance extraction, our data matrix was derived from Andy Carlson's all-pairs data set [5]. This time we didn't have any noun phrases to begin with so we need to sample them from the all-pairs data set as well. Heuristically, we concluded that the majority of the meaningful noun phrase pairs were formed by title case and uppercase phrases so we sampled the data in a way which we keep such entries. The resulting dataset consisted of 13,020,010 unique noun phrase pairs and 11,424,413 unique contextual patterns. They are stored in our traverse-efficient array-based sparse matrix M_{ij}, with each cell corresponding to the co-occurrence number of contextual pattern $\#i$ against noun phrase pair $\#j$. The matrix was loaded to the memory in advance of the execution.

Data Preparation Procedure

The data matrix was produced as a result of the following steps:

1. From the file of relational contexts (F_5), we extracted the full list of context literals, sorted them lexicographically and eliminated the duplicates. The results were stored in file *CONTS.txt*
2. From the file of pair-context co-occurrence (F_6), we extracted the full list of paired noun phrase literals, sorted them lexicographically and eliminated duplicates. The results were stored in file *NPAIRS.txt*
3. In file F_6, we sorted every line (lists of co-occurring contexts for a given noun phrase pair) in itself, lexicographically. The resulting file was *COOCS.txt*
4. We split *COOCS.txt* into 252 pieces, with names *COOCS_i.txt* to be sorted. This was so that the files could fit in the memory entirely, for sorting.
5. We sorted every *COOCS_i.txt* file by their lines (i.e. by the noun phrase pairs at the beginning of the lines)
6. We merged *COOCS_i.txt* 252 ways. The resulting file was a giant 40GB file, with contexts ordered in-line and noun phrase pairs ordered across lines.
7. We subsampled *NPAIRS.txt* to uppercase & titlecase noun phrase pairs.
8. We subsampled *COOCS.txt* to uppercase & titlecase noun phrase pairs.

It took us roughly 24 hours to execute all these steps accumulatively. The final data set is made up of the following 3 files:

- *"CONTS.txt"*. This file contains the lexicographically sorted list of 11,424,413 relational contexts in the universe $\mathcal{R}$.
- *"NPAIRS.txt"*. This file contains the lexicographically sorted list of 13,020,010 noun phrase pairs in the universe $\mathcal{T}$ such that both noun phrases begin with a capital letter.
- *"COOCS.txt"*. This file contains the co-occurrence data for all the pairs and contexts declared in the two files above, accommodating $\approx$ 48,800,000 nonzero co-occurrence entries.

These 3 files can now be directly used by our program to populate our *pair* $\times$ *context* data matrix.

6.2.3 Experimental Configuration

We used 10 relations, each initialized with 5-6 seeds. The program ran for 10 iterations. Experimental configuration was similar to our previous categorical extraction experiments. We promoted the top 5 new noun phrases for every relation as trusted, which were to be used as seeds in subsequent rounds. As before, we set the tolerance threshold $\theta = 50\%$ as it produced the most semantically accurate tolerance classes. We also set $\alpha = 0.5, \beta = 0.25$, and $\gamma = 0.25$.

6.2.4 Evaluation Criteria

In order to measure the performance, we took two different approaches both of which involved the Precision@30 metric:

1. To get a ranking-based result, we followed the same convention as CBS [39]: In any iteration, after noun phrases are scored and ranked for a relation, the percentage of the correct pairs in the set of the top 30-ranked pairs is calculated.

2. To get a promotion-based result, we followed the steps of CPL [3]: From the set of all promoted pairs for a given relation, we sampled 30 pairs to be evaluated and we calculated the percentage of the correct pairs within that set.

The correctness of the pairs are judged by ourselves, as before.

6.2.5 Results

On a Windows 7 machine with 2.40 GHz Intel i7 processor and 16GB memory, it took about 1 hour and 10 minutes to run our test scenario in single thread. Table 6.7 summarizes the results, which are mostly self-evident. It can be observed that for most relations, TPL performed high precision extractions throughout the iterations, steering clear of the semantic drift problem. Average precision values are comparable with CPL and are promising for both metrics.

To elaborate some details, Table 6.8 shows the promotions in the last iteration. First of all, we should note that our data set was not up to date and we did not flag expired events. We accepted a pair for a relation if it has ever been correct. For example, athlete *Taurasi* used to play for team *Phoenix* so it was accepted. We also accepted properly formed aliases e.g. *Pompey* & *Portsmouth F.C*, *L.A.* & *Los Angeles* and likewise.

Table 6.7: Precision@30 results of TPL and CPL as seen in [3] (%).

Evaluation	**Ranking-based**			**Promotion-based**			
	TPL			**TPL**			**CPL**
Iterations	**1**	**5**	**10**	**1**	**5**	**10**	**10**
Relations							
Athlete-Team	100	90	87	100	96	87	100
CEO-Company	100	100	100	100	100	100	100
City-Country	100	100	100	100	100	100	93
City-State	100	100	100	100	100	100	100
Coach-Team	93	93	93	100	100	93	100
Company-City	83	90	93	40	84	97	50
Stadium-City	97	93	80	80	92	70	100
State-Capital	100	97	73	100	100	63	60
State-Country	100	100	100	100	100	100	97
Team-vs-Team	93	83	80	100	84	80	100
Average	96.6	94.6	90.6	92.0	95.6	89.0	90.0

In Table 6.8, we can observe different types of mis-promotions. One of them is improperly tokenized pairs, such as (*Grant, Former Chelsea*) which is caused by *Grant* being the former *Chelsea* football team coach. Such cases are hard to identify and easy to deceive the algorithm since contextually, it appears as if the name of the team is "*Former Chelsea*". Other erroneous promotions are somewhat off-course pairs, which are relevant in the context but not properly fit the definition. *State-Capital*, one of the few poorly performed relations, demonstrates this case. The incorrect promotions are formed by states and non-capital cities, however, the cities in question are indeed located in the respective states. For that particular relation, we observed drift from *Capital* to *City*. Nonetheless, that relation challenged CPL as well [3], which suggests, it was a difficult one to work on.

6.2.6 Discussion

Based on the experiments, we observed that our assumptions regarding the similarity of the categorical and relational information labeling problems were true. These two problems indeed responded to the same theoretical approach in a similar manner. For most relations, TPL maintained high quality extractions and high precision values throughout the iterations, steering clear from the concept drift problem. The clarity of the concepts and inter-intra class similarity were once again determining factors of the correction of the extractions.

Compared to the category learner module of TPL, one big challenge was the exploding size of entities. For categorical information extraction, we needed to work on $|\mathcal{N}|$ noun phrases. The universe of noun phrase pairs, however, is bound by the Cartesian Square ($\mathcal{T} \subseteq \mathcal{N}^2$) so the number of pairs required to create a reasonable subsample is much larger for a relation learner. Not to mention the more noun phrase pairs implies the more contextual patterns, which results in a much larger co-occurrence matrix. This is why we had to work on a 13 million by 11 million matrix for the relation learner as opposed to 69 thousand by 60 thousand for our category learner. Fortunately, these matrices are sparse. Moving to a compiled language and implementing an efficient array based sparse matrix helped us boost the execution time. At the end, it even outperformed our category learner.

We sustained our simplistic approach to form the relation learner. Though this simple approach seems to perform well, the accuracy of the extractions may be open for improvement with the help of additional constraints. One constraint is type-checking. If the relation learner is combined with the category learner, they can help each other by "testifying." For instance, when learning the pair (*Winnipeg, Canada*) for *City-Country*, we can check if *Winnipeg* qualifies as a *City* and if *Canada* qualifies as a *Country*. One other constraint is mutual exclusion, as pointed out before. We can define non-overlapping relations as mutually exclusive and check if a given pair qualifies for any excluded category. Nell makes active use of these constraints [3] and they are indeed shown to be effective.

6.3 Complexity and Scalability Issues

There are several parameters that contribute to the time complexity of the TPL algorithm including number of iterations $\#_{iter}$, number of categories (or relations) $\#_{cat}$, number of promotions per category $\#_{prom}$, number of noun phrases (or pairs) $|\mathcal{N}|$ and number of contexts $|\mathcal{C}|$ in the universe. The most expensive part is calculating the lower and upper approximations for the trusted nouns, which requires the calculation of the tolerance classes for the co-occurring contexts of those nouns. Provided that $\#_{iter}$, $\#_{cat}$ and $\#_{prom}$ are fixed, the execution time would depend on the sparsity of the data matrix, which can be ensured by preprocessing the matrix to filter out the low cardinality co-occurrence values (i.e. only keep the top n co-occurrence values). This provides an upper bound on the matrix density and it is indeed how we formed our matrix. Ultimately, the complexity for TPL becomes $O(|\mathcal{C}|)$ matching the $O(N =$

$\#_{terms}$) value of TRSM-based document clustering algorithm [11]. It is not surprising since both methods employ the tolerance rough set framework. The space complexity is drawn by the total number of non-zero co-occurrence values which is limited by $O(|\mathcal{N}| * |\mathcal{C}|)$ for the unfiltered data matrix and $O(|\mathcal{C}|)$ for the filtered.

Although we only tested TPL in a single-thread setup on CPU, we believe it is a highly scalable algorithm. Most tasks are independent of one another and can be parallelized on various levels. For instance, each category can be processed in parallel for its current iteration, each trusted noun can be processed in parallel for its approximations and each context can be treated likewise for its tolerance class. This should also mean that the system can be deployed on GPU to facilitate parallel processing.

Table 6.8: Pairs promoted by TPL in iteration 10. Wrong extractions are italicized.

Athlete-Team	**CEO-Company**
Diana Taurasi , Ph'x Mercury	Vittorio Colao , Vodafone
Jordin Tootoo , Nashville	Vijay Mallya , Kingfisher
Stanley Robinson , UConn	John Gay , Gunns
Jason Richardson , Golden St.	Adam Pearson , Derby
James Posey , Heat	Peter Storrie , Pompey
City-Country	**City-State**
Managua , Nicaragua	Akron , Ohio
Seoul , South Korea	Billings , Montana
Kampala , Uganda	San Antonio , Texas
Kingston , Jamaica	Colorado Springs , Color'o
Nassau , Bahamas	Burlington , Vermont
Coach-Team	**Company-City**
Tony Smith , England	Chipotle , Denver
Eriksson , Former England	SGI , Mountain View
Zico , Fenerbahce	Sega , Tokyo
Roy Hodgson , Finland	XM Sat. Radio , Wash'n
Grant , Former Chelsea	Vodafone , Newbury
Stadium-City	**State-Capital**
Dodger , L.A.	*Ohio , Cincinnati*
Dolphin , South Florida	*Arizona , Tucson*
Drillers , Tulsa	Connecticut , Hartford
Kings Park , Durban	*Michigan , Detroit*
El Madrigal , Villarreal	*Vermont , Burlington*
State-Country	**Team-vs-Team**
Kansas , United States	XSV , Dynasty
Colorado , US	GT , FSU
Connecticut , U.S.	*Worcester , Georgia*
Oregon , United States	PSG , Lens
Minnesota , United States	Tottenham , Man City

Chapter 7

Conclusion

In this thesis, we considered a particular type of web information labeling problem: populating categorical and relational facts as noun phrases and noun phrase pairs by using the contextual co-occurrence statistics in a dataset harvested from web documents. We proposed a granular-based model rooted in tolerance rough sets, adapting the document representation model TRSM to this specific problem by representing noun phrases and noun phrase pairs as granular sets of their contextual patterns and investigating their similarity by means of their representation. We also provided a practical solution; we introduced a semi-supervised algorithm, Tolerant Pattern Learner (TPL), which employs the proposed noun-context and pair-context representation model to iteratively populate instances for categories and relations. We have demonstrated that the TPL algorithm produces promising results for both streams of facts, which is comparable with the benchmark algorithms CBS and CPL in terms of precision.

7.1 Future Research Directions

Here are some possible future research directions on this work:

- **Ensemble learning** We may combine categorical and relational modules of TPL to create an ensemble learner, which can also employ type-checking and mutual exclusion constraints to provide further justification for the candidates, aiding the learning process.

- **Dynamic ontology discovery** We may consider developing a tolerance rough

set based model that works on an evolving ontology, by discovering categories and relations on-the-fly.

- **Temporal Scoping** One may also consider examining the temporal scope of the facts learned by TPL, and introduce a chronological order, as done by Talukdar et al. [38, 37].

Appendix A

Snapshots from the Experiments

In this chapter, we present some snapshots from our experiments in order to provide the reader a better illustration of our experimental environment.

A.1 Categorical Extraction

In Figure A.1 we see a snapshot from the execution of our TPL category instance extractor module we described in Section 6.1. The driver function Run_TPL takes 3 parameters. W is the list of noun phrase literals (i.e. $\mathcal{N}$), C is the list of contextual pattern literals and D is the co-occurrence matrix (see Figure A.2 for their contents). In that snapshot, each seed for the category *Company* is lower and upper approximated by means of tolerance rough sets.

Figure A.3 shows a full iteration on the category *Sport*. We first calculate the enhanced representation of the seed nouns and then we calculate the macro score for every noun phrase in W, by using Eq. 5.10. We than rank the noun phrases by those score and dump the top-30 to the screen, for Precision@30 evaluation. At the end, we promote the 5 new top-ranked instances as seeds for the upcoming iterations.

A.2 Relational Extraction

In Figure A.4 we see a snapshot from the execution of our TPL relation instance extractor module we described in Section 6.2. The executable requires three input files, "NPAIRS.txt", "CONTS.txt", and "COOCS.txt". (Their content is illustrated in Figure A.5.) Similarly to the previous case, the seeds pairs of the relation *City-*

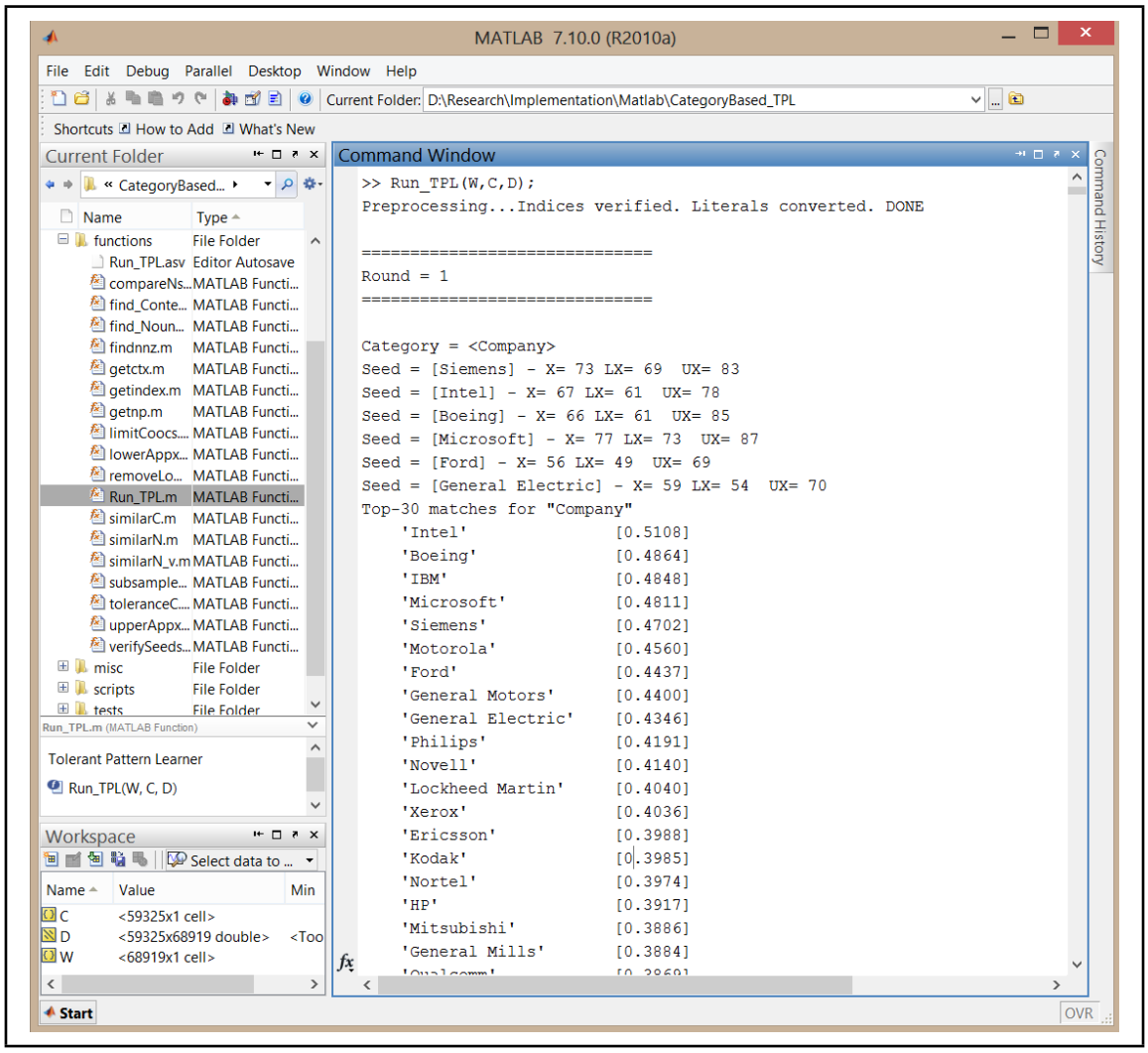


Figure A.1: TPL category extractor in action.

State are processed for enriched representation and they are used to calculate for every noun phrase pair in the universe, by using Eq. 5.11. The pairs are than ranked and the top-scoring 30 pairs are dumped to the screen, for Precision@30 evaluation. Finally, the 5 top-ranked new pair is promoted to be used as seeds.

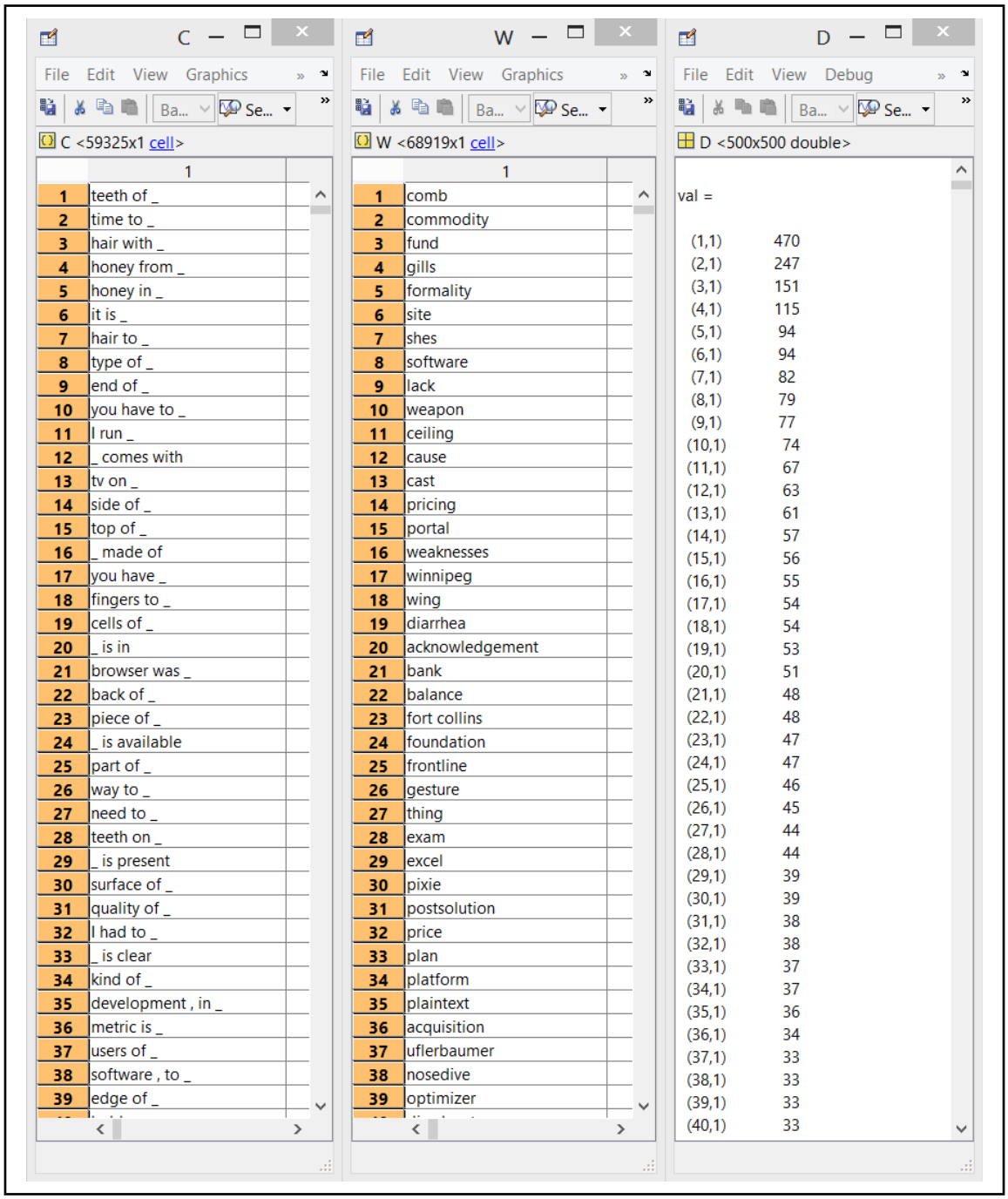


Figure A.2: Input data C, W, D for the category learner. D is a $|C| \times |W|$ sparse matrix and it is sub-sampled to 500×500 for display purposes.

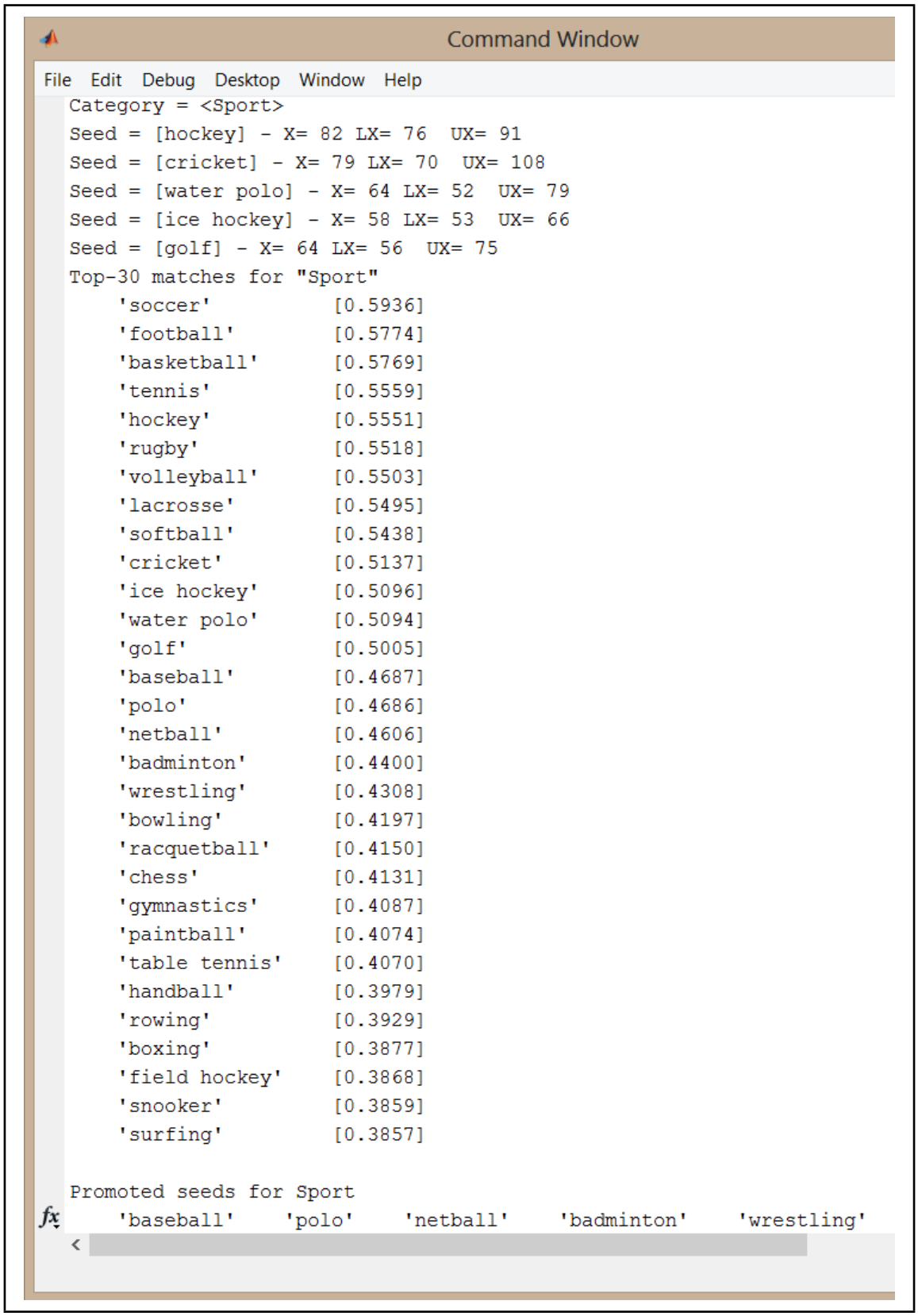


Figure A.3: Ranking and labeling phrases for category “Sport”.

Figure A.4: TPL relation extractor in action: Populating the relation “City-State”.

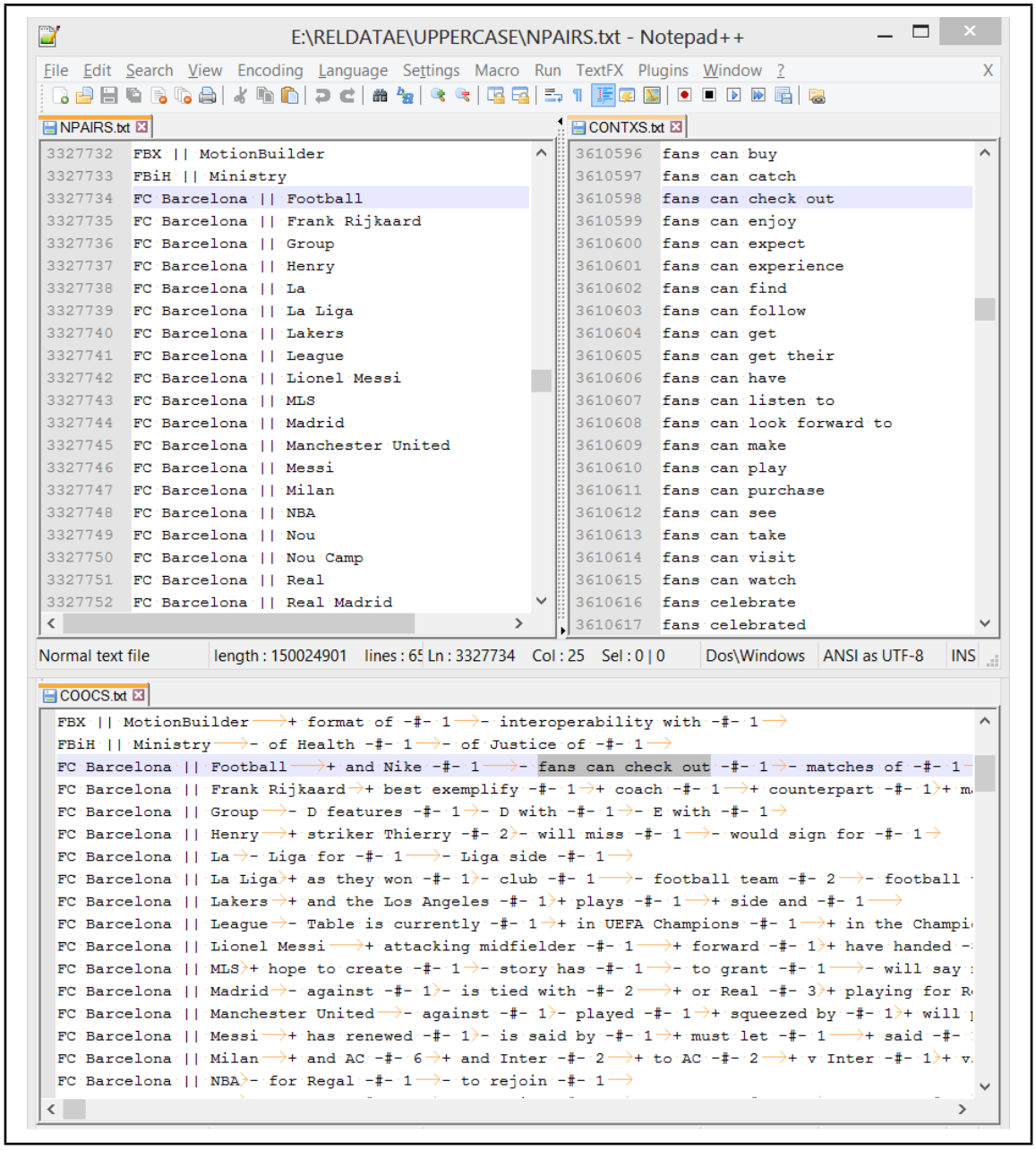


Figure A.5: Three input files for the relation learner, as discussed in Section 6.2.2. In "COOCS.txt" the (+) or (−) sign denotes whether the pair is in true or inverted order for the succeeding context.

Bibliography

[1] M. Banko, M.J. Cafarella, S. Soderland, M. Broadhead, and O. Etzioni. Open Information Extraction from the Web. In *Proceedings of IJCAI*, pages 2670–2676, 2007.

[2] Jamie Callan and Mark Hoy. Clueweb09 Data Set, 2009.

[3] A. Carlson, J. Betteridge, R. C. Wang, E. R. Hruschka, Jr., and T. M. Mitchell. Coupled Semi-supervised Learning for Information Extraction. In *Proceedings of the Third ACM International Conference on Web Search and Data Mining*, pages 101–110, 2010.

[4] Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R. Hruschka, and Tom M. Mitchell. Toward an Architecture for Never-ending Language Learning. In *Proceedings of the Twenty-Fourth Conference on Artificial Intelligence, AAAI 2010*, volume 2, pages 1306–1313, 2010.

[5] Andy Carlson. All-pairs Data Set, 2010.

[6] Olivier Chapelle, Bernhard Schölkopf, Alexander Zien, et al. *Semi-supervised Learning*, volume 2. Cambridge: MIT press, 2006.

[7] J.R. Curran, T. Murphy, and B. Scholz. Minimising Semantic Drift with Mutual Exclusion Bootstrapping. In *Proc. of PACLING*, pages 172–180, 2007.

[8] Oren Etzioni, Michael Cafarella, Doug Downey, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S. Weld, and Alexander Yates. Unsupervised Named-entity Extraction from the Web: An Experimental Study. *Artif. Intell.*, 165(1):91–134, June 2005.

[9] Anthony Fader, Stephen Soderland, and Oren Etzioni. Identifying Relations for Open Information Extraction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 1535–1545, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics.

[10] Zoubin Ghahramani and Katherine A. Heller. Bayesian Sets. In *in Advances in Neural Information Processing Systems*, volume 18, 2005.

[11] Tu Bao Ho and Ngoc Binh Nguyen. Nonhierarchical Document Clustering Based on a Tolerance Rough Set Model. *International Journal of Intelligent Systems*, 17:199–212, 2002.

[12] Saori Kawasaki, Ngoc Binh Nguyen, and Tu Bao Ho. Hierarchical Document Clustering Based on Tolerance Rough Set Model. In *Proceedings of the 4th European Conference on Principles of Data Mining and Knowledge Discovery*, pages 458–463, 2000.

[13] Jayant Krishnamurthy and Tom Mitchell. Vector Space Semantic Parsing: A Framework for Compositional Vector Space Models. In *Proceedings of the ACL 2013 Workshop on Continuous Vector Space Models and their Compositionality*, pages 1–10, Sofia, Bulgaria, August 2013. Association for Computational Linguistics.

[14] Lokesh Kumar and Parul Kalra Bhatia. Text Mining: Concepts, Process and Applications. *Journal of Global Research in Computer Science*, 4(3):36–39, 2013.

[15] Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N. Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick van Kleef, Sören Auer, and Christian Bizer. DBpedia - A Large-scale, Multilingual Knowledge Base Extracted from Wikipedia. *Semantic Web Journal*, 2014.

[16] S. Marcus. Tolerance Rough Sets, Cech Topologies, Learning Processes. *Bull. Polish Academy of Sciences, Technical Sciences*, 42(3):471–487, 1994.

[17] John Markoff. Start-Up Aims for Database to Automate Web Searching. `http://www.nytimes.com/2007/03/09/technology/09data.html?_r=0`. New York Times. Accessed: 2014-10-15.

[18] Jack Menzel. Deeper Understanding with Metaweb. `http://googleblog.blogspot.ca/2010/07/deeper-understanding-with-metaweb.html`. Google Official Blog. Accessed: 2014-10-15.

[19] Thahir Mohamed, Estevam Hruschka, and Tom Mitchell. Discovering Relations Between Noun Categories. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1447–1455, Edinburgh, Scotland, UK., July 2011. Association for Computational Linguistics.

[20] Chi Lang Ngo. A Tolerance Rough Set Approach to Clustering Web Search Results. Master's thesis, Warsaw University, 2003.

[21] J. Nieminen. Rough Tolerance Equality and Tolerance Black Boxes. *Fundamenta Informaticae*, 11:289–296, 1988.

[22] Zdzislaw Pawlak. Rough Sets. *International Journal of Computer & Information Sciences*, 11(5):341–356, 1982.

[23] Zdzislaw Pawlak. *Rough Sets: Theoretical Aspects of Reasoning About Data*. Kluwer Academic Publishers, Norwell, MA, USA, 1992.

[24] Zdzislaw Pawlak and Andrzej Skowron. Rudiments of Rough Sets. *Information Sciences*, 177(1):3–27, 2007.

[25] Witold Pedrycz, Andrzej Skowron, and Vladik Kreinovich. *Handbook of Granular Computing*. Wiley-Interscience, New York, NY, USA, 2008.

[26] James Peters and Piotr Wasilewski. Tolerance Spaces: Origins, Theoretical Aspects and Applications. *Information Sciences*, 195(1-2):211–225, 2012.

[27] L. Polkowski, A. Skowron, and J. Zytkow. Tolerance Based Rough Sets. In *Lin, T.Y., Wildberger, M. (eds.) Soft Computing: Rough Sets, Fuzzy Logic, Neural Networks, Uncertainty Management, Knowledge Discovery*, pages 55–58, San Diego, 1994. Simulation Councils Inc.

[28] Alan Ritter, Mausam, Oren Etzioni, and Sam Clark. Open Domain Event Extraction from Twitter. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '12, pages 1104–1112, New York, NY, USA, 2012. ACM.

[29] M. Schroeder and Wright M. Tolerance and Weak Tolerance Relations. *Journal of Combinatorial Mathematics and Combinatorial Computing*, 11:123–160, 1992.

[30] AI Computer Science and University of Washington Engineering. Visual for Unstructured Text to Structured Text. `http://ai.cs.washington.edu/www/media/icons/pubs/Screen_shot_2010-09-11_at_9.36.44_AM.png`. Accessed: 2014-09-17.

[31] Cenker Sengoz and Sheela Ramanna. A Semi-supervised Learning Algorithm for Web Information Extraction with Tolerance Rough Sets. In *AMT 2014*, pages 1–10, 2014.

[32] Cenker Sengoz and Sheela Ramanna. Learning Relational Facts from the Web: A Tolerance Rough Set Approach. *Pattern Recognition Letters*, 2015. (Submitted).

[33] Lei Shi, Xinming Ma, Lei Xi, Qiguo Duan, and Jingying Zhao. Rough Set and Ensemble Learning Based Semi-supervised Algorithm for Text Classification. *Expert Syst. Appl.*, 38(5):6300–6306, May 2011.

[34] Andrzej Skowron and Jaroslaw Stepaniuk. Tolerance Approximation Spaces. *Fundam. Inf.*, 27(2,3):245–253, August 1996.

[35] T. Sørensen. *A Method of Establishing Groups of Equal Amplitude in Plant Sociology Based on Similarity of Species Content and Its Application to Analyses of the Vegetation on Danish Commons*. Biologiske skrifter. I kommission hos E. Munksgaard, 1948.

[36] Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. Yago: A Core of Semantic Knowledge. In *16th International World Wide Web Conference (WWW 2007)*, New York, NY, USA, 2007. ACM Press.

[37] Partha Pratim Talukdar, Derry Wijaya, and Tom Mitchell. Acquiring Temporal Constraints Between Relations. In *Proceedings of the Conference on Information and Knowledge Management (CIKM 2012)*, pages 992–1001, October 2012.

[38] Partha Pratim Talukdar, Derry Wijaya, and Tom Mitchell. Coupled Temporal Scoping of Relational Facts. In *Proceedings of the Fifth ACM International Conference on Web Search and Data Mining (WSDM)*, pages 73–82, February 2012.

[39] S. Verma and E. R. Hruschka, Jr. Coupled Bayesian Sets Algorithm for Semi-supervised Learning and Information Extraction. In *ECML PKDD Part II LNCS 7524*, pages 307–322, 2012.

[40] Gloria Virginia and Hung Son Nguyen. Lexicon-based Document Representation. *Fundam. Inform.*, 124(1-2):27–46, 2013.

[41] P. Wasilewski, J.F. Peters, and S. Ramanna. Perceptual Tolerance Intersection. In *LNCS 6086*, pages 277–286. Springer, 2010.

[42] Derry Wijaya, Ndapa Nakashole, and Tom Mitchell. CTPs: Contextual Temporal Profiles for Time Scoping Facts via Entity State Change Detection. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, Doha, Qatar., October 2014. Association for Computational Linguistics.

[43] Derry Tanti Wijaya and Philip Gianfortoni. "Nut Case: What Does It Mean?": Understanding Semantic Relationship Between Nouns in Noun Compounds Through Paraphrasing and Ranking the Paraphrases. In *Proceedings of the 1st International Workshop on Search and Mining Entity-relationship Data*, SMER '11, pages 9–14, 2011.

[44] Derry Tanti Wijaya and Reyyan Yeniterzi. Understanding Semantic Change of Words over Centuries. In *Proceedings of the 2011 International Workshop on Detecting and Exploiting Cultural Diversity on the Social Web*, DETECT '11, pages 35–40, New York, NY, USA, 2011. ACM.

[45] Fei Wu and Daniel S. Weld. Open Information Extraction Using Wikipedia. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, pages 118–127, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

[46] J.T. Yao, A.V. Vasilakos, and W. Pedrycz. Granular Computing: Perspectives and Challenges. *IEEE Transactions on Cybernetics*, 43(6):1977–1989, 2013.

[47] L.A. Zadeh. Towards a Theory of Fuzzy Information Granulation and Its Centrality in Human Reasoning and Fuzzy Logic. *Fuzzy Sets Systems*, 177(19):111–127, 1997.