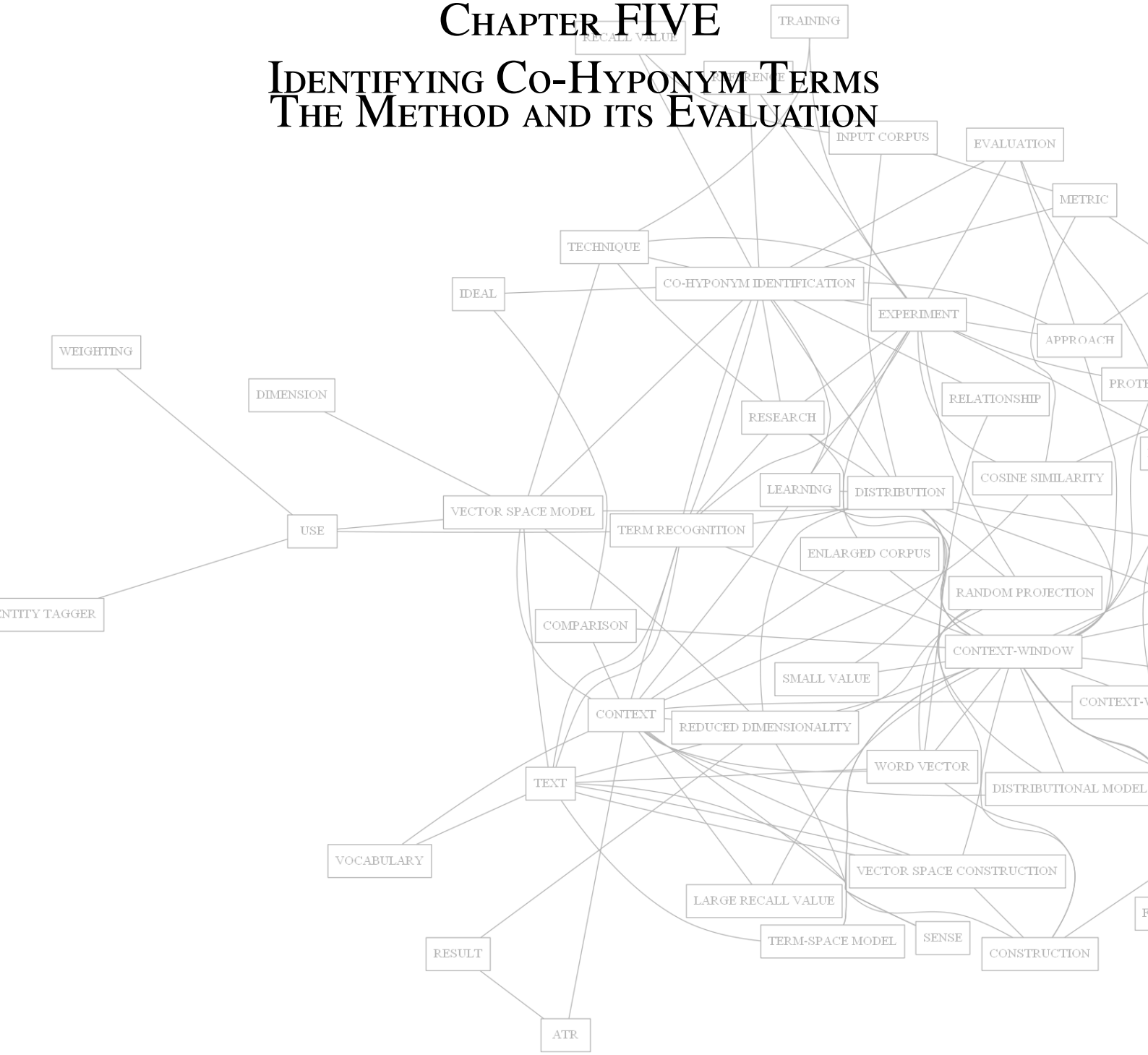
Chapter FIVE
Identifying Co-Hyponym Terms The Method and its Evaluation



 Corresponding author: behrangatoffice[at]gmail.com. This document is a chapter in QasemiZadeh (2015). Cite as: Behrang QasemiZadeh (2015). Investigating the Use of Distributional Semantic Models for Co-Hyponym Identification in Special Corpora. PhD Thesis, NUI Galway.

This page is intentionally left blank.

Contents

This page is intentionally left blank.

List of Figures

List of Tables

This page is intentionally left blank.

Chapter 5

Identifying Co-Hyponym Terms: The Method and its Evaluation

In this chapter, the proposed method for identifying co-hyponym terms is explained and evaluated. The principles of automatic term recognition and distributional semantics are combined to implement a method that extracts terms from a category of similar concepts (i.e., co-hyponyms). After the extraction of candidate terms, stable random projections are employed to represent these candidate terms as low-dimensional vectors. These vectors are derived automatically from the co-occurrences of candidate terms and words that appear in their proximity (context-windows). In a memory-based k-nearest neighbours learning framework, and using a small set of manually annotated terms, co-hyponym terms are identifiedf by classifying these vectors.

Section 5.1 reintroduces the task and justifies the proposed method based on the principles of distributional semantics. This introduction is followed by delineating the method in Section 5.2. In Section 5.3, the evaluation framework and material are discussed. Results from a number of experiments in the defined evaluation framework are reported in Section 5.4 and discussed in Section 5.5. After suggesting an approach for improving the performance of the method for large recall values in Section 5.6, the chapter concludes with a summary in Section 5.7.[1]

[1]The proposed method in this chapter has been published and evaluated partly in Zadeh and Handschuh (2014a) and Zadeh and Handschuh (2014b).

5.1 Introduction

As is explained in Chapter 3, in automatic term recognition (ATR), given a special corpus, the goal is to automatically extract a specialised vocabulary—that is, in its simplest form, a terminological resource composed of a set of lexical units known as *terms*. Terms can be either *simple* or *complex*—that is, single-token or multi-token lexical units. A terminological resource is an indispensable component of a system employed to communicate a specialised knowledge. Hence, it signifies diverse concepts in the targeted domain knowledge. These concepts, and thus terms, are often organised according to a classification scheme, which is determined by a number of factors such as the intended application context. In an information system, this categorisation is often a major mechanism to reflect the structure of the (conceptualised) specialised knowledge—for example, as practised in *ontology engineering* (e.g., see L'Homme and Bernier-Colborne, 2012, for an overview) and as explained in Section 1.1.

For instance, the terms *lexicon*, *corpus*, *terminology*, *parsing* and *information extraction* are conceivable entries from a terminological resource in the domain of computational linguistics. In this list, *lexicon*, *parsing*, and *terminology* are simple terms, whereas *information extraction* is a complex term. According to a classification scheme (i.e., a conceptualisation of the domain knowledge), *lexicon* and *corpus* can be grouped under the concept category *language resource*. Similarly, *information extraction* and *parsing* can be classified under the category *technology and process* (Figure 5.1). It is worth mentioning that a term can appear in more than one category of concepts. In the given example, the term *terminology* appears in both categories, as a term that can signal both a language resource and a processing resource (see also Figure 3.2 in Chapter 3). Accordingly, as discussed in Section 1.1, terms under each category of concepts are in a co-hyponymy relationship since they share a similar hypernym. For instance, in the example given in Figure 5.1, *lexicon*, *terminology* and *corpus* are *co-hyponym terms*.

A number of research studies have attempted to extract and define a scheme for the categorisation of terms into co-hyponym groups, either implicitly using a clustering technique (e.g., as suggested in Dupuch et al., 2014; Cimiano et al., 2005), or explicitly by inducing inference rules—such as using an automatic or manual engineering of Hearst's (1992) lexico-syntactic patterns (e.g., as suggested in Maynard et al., 2009).[1] In a large number of applications, however, the classification scheme[2] is known (or, at least, a partial knowledge of it exists). In this case, finding co-hyponym terms that belong to a particular category of concepts is a typical task. In the context of ontology engineering, the former research is usually a sub-process of the *ontology learning* task, whereas the latter is often demanded for *ontology population* (see Buitelaar et al., 2005; Wong et al., 2012). The focus in this Chapter is on the latter.

Entity extraction methods are commonly employed to distil co-hyponym terms, of which *bio-entity recognition tasks* are the most established examples (e.g., see Kim et al.,

[1] Note that the use of these patterns is not limited to taxonomy induction processes, as is shown in the next few pages.

[2] That is, the set of hypernyms in the conceptualisation of the domain knowledge under investigation.

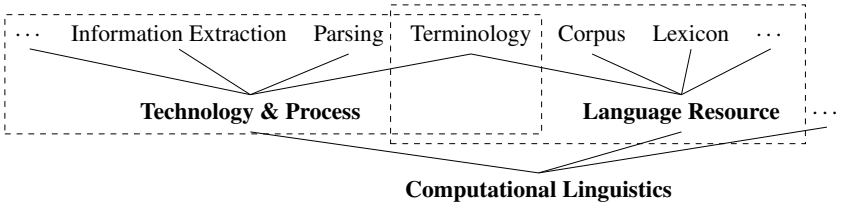


Figure 5.1: Taxonomy and co-hyponyms: This example shows a simple taxonomy in the domain of computational linguistics. Terms are classified into two categories: *language resource* and *technology and process*. Terms under each category (placed in boxes) form a group of *co-hyponyms*. A number of terms such as *terminology* in this example can be *polysemous*, hence classified under more than one category of concepts. An ideal ATR system extracts terms listed in the top row of this figure. As suggested, terms can be organised according to a taxonomy. One way to approach this task is to identify co-hyponyms.

2004). As detailed in Section 1.2, these methods, however, are not suitable for a number of use cases due to their lack of flexibility and a mechanism for resembling the knowledge structure. Moreover, developing entity taggers is restricted by the availability of manually annotated corpora. In these corpora, individual mentions of terms and their concept category are required to be manually annotated. A few techniques exploit information redundancy in very large corpora to obviate this requirement.[1] However, in special (domain-specific) corpora, using information redundancy alone can be insufficient to automatically generate annotated data (e.g., as shown in Section 5.4.1.1). Last but not least, using an entity tagger for extracting co-hyponyms abandons an important characteristic of special corpora—that is, reduced lexical ambiguity.

As discussed in Chapter 1 and 3, in specialised languages, terms are often coined to facilitate communication by reducing lexical ambiguity. Therefore, synonymy and polysemy are less frequent in specialised languages than general language. The concept of *sense* can be defined differently depending on the context (e.g., see Cimiano et al., 2013, for an elaboration of a three faceted definition in the *ontology-lexicon* framework).[2] Thus, the meaning of *polysemy* can be interpreted differently. In the context of this discussion, I suggest that a domain ontology populated by instances extracted from a special corpus plays an analogical role to that of a lexical database supplemented by word senses in general language. Accordingly, I assume that the relationships between instances (i.e., terms in the special corpus) and concepts in the domain ontology is similar to the relationship between word forms and senses in a general language lexical database.[3] Let me explain the proposed argument by a comparison between WordNet and the *GENIA ontology* populated by annotations provided in the GENIA corpus.

[1] For instance, see Etzioni et al. (2005). These methods are often employed for the extraction of proper nouns in general language. Usually, the manual annotation is replaced by hand-crafted lexico-syntactic patterns, or a small number of *seed examples*.

[2] *Sense* has different senses and thus is polysemous, so to speak!

[3] The conceptualisation behind a domain ontology (i.e., the number of classes and their relationship) plays a role in the proposed analogy and the subsequent proposed comparison in this section (i.e., the relationship between *granularity* vs. *condensation* of concepts in domain ontologies). For simplicity without the loss of generality, I discard this relationship.

WordNet is unarguably a general language vocabulary in which the proportion of polysemous words is approximately 17% (according to Miller, 1995). The GENIA corpus (which is a well-known corpus in the domain of molecular biology) provides manual annotations for 92,722 term mentions (Kim et al., 2003). These term annotations in the GENIA corpus are grounded on the GENIA ontology. The GENIA ontology consists of 45 classes that are organised in a hierarchical taxonomy of 6 levels. The annotated 92,722 term mentions form a vocabulary of 34,077 distinct entries (hereinafter *GENIA terminological resource*). Among them, individual mentions of 1,373 entries are annotated with at least two classes from the GENIA ontology. If these terms are considered polysemous, compared to WordNet, only a small fraction (i.e., $\frac{1372}{34077} = 4\%$) are polysemous.[1] Although a direct comparison of the two resources can be not accurate,[2] it is still a reliable evidence of the expected differences of the properties[3] of relationships between entries in a terminological resource and a general language lexical database. Disregarding the employed vocabulary for describing this phenomenon (i.e., whether or not to use the word *polysemy*), this thesis exploits the described phenomenon and suggests a method to identify co-hyponym terms.

Similar to ATR and in contrast to entity recognition tasks, the method proposed for identifying co-hyponym terms works at a corpus level and does not deal with individual occurrences of a term in text snippets. However, in contrast to ATR (which extracts terms from diverse categories of concepts in a domain knowledge) and similar to entity recognition, the objective is to extract a particular subset of terms that signify a similar hypernym.

The proposed method in the investigated use case has many practical applications: ranging from classic applications in information retrieval (e.g., see principles that are suggested by Rijsbergen, 1977, for *index term weighting*) to more recent so-called *ontology-based information systems* as (assistive) tools for maintaining and populating domain ontologies. Apart from these two broad applications, there is a growing demand of information extraction tasks that, in fact, can be boiled down to the proposed co-hyponym identification task. For example, in the so-called expertise finding task (e.g., see Balog and de Rijke, 2008) a major process is the identification of *expertise topics* (e.g., Buitelaar and Eigner, 2008). In this scenario, the *expertise topics* are, in fact, a set of co-hyponym terms, and can thus be identified using the method proposed in this chapter. Classifying user-generated annotations in applications such as *tag-based information access* (e.g., Yi, 2010) is another example. A similar use case is presented by Chakraborty et al. (2014) for extracting information from unstructured text ads. Last but not least are applications such

[1] A same conclusion can be drawn by analysing the distribution of words senses in SemCor (Mihalcea, 1998)—that is, a corpus of general language text annotated with WordNet senses—and the GENIA corpus.

[2] For instance, as they are built from opposite viewpoints. In constructing WordNet, for a given word, an inventory of all the meanings of the word is made by searching the occurrences of the word in large text corpora. In the GENIA terminological resource, however, from a limited number of observations in the specialised corpus, all the concepts that terms represent are collected. In the proposed example, as put by Cimiano et al. (2013), a *reification* of the link between terms (lexical forms) and ontological references from the GENIA corpus are assumed to represent senses. According to this terminology, in this thesis, terms that are reified to the same ontological reference are considered co-hyponyms.

[3] Such as diversity and frequency.

as *technology watch* that intend to provide *technological intelligence* by machine reading of large text corpora (e.g., as explained in QasemiZadeh, 2010).

As described in Section 1.2, the co-hyponym identification task can be formulated as a classification task (see Figure 1.1 in Chapter 1). Therefore, the proposed method is realised as an ad hoc term-weighting procedure on top of an ATR system's two-step procedure—candidate term extraction followed by term weighting and ranking. As described in Chapter 3, after the extraction of candidate terms, ATR combines the *unithood* and *termhood* scores to weight terms (Figure 5.2). Unithood characterises the strength of syntagmatic relationships between the tokens that compose complex terms. Termhood, however, characterises a paradigmatic relationship—that is, the association of candidate terms to the concepts in a specialised knowledge domain, which is verbalised by the special corpus under investigation. Termhood in ATR disregards terms' associations to different concept categories. In contrast, in the proposed task, a score that discerns these associations must be devised. This score, however, is similar to termhood in the sense that it characterises a *paradigmatic* relationship—that is, the co-hyponymy relationship between terms that are grouped under a category of concepts, such as the relationship between *lexicon*, *corpus*, and *terminology* exemplified in Figure 5.1.

A distributional approach is employed to design this score. By extending Harris's (1954) distributional hypothesis, one can claim that the context in which terms are used can be exploited to identify their concept category.[1] Hence, in this thesis, it is assumed that the association of a term to a concept category can be characterised using the syntagmatic relation of the term and its co-occurred words in windows of text extended in the vicinity of the term's mentions in the corpus (i.e., *context-windows* as shown in Figure 5.3).[2] Accordingly, I hypothesise that co-hyponym terms tend to have similar distributional properties in context-windows. In order to quantify these distributional similarities, vector space models—which are described thoroughly in Chapter 2—are employed.

Words that appear in context-windows are represented by the elements of the standard basis of a vector space—that is, informally, dimensions of a vector space—and each candidate term is represented by a vector. In this vector space, the coordinates of vectors is determined by the co-occurrence frequency of words that appear in context-windows and candidate terms in a special corpus. Consequently, the values assigned to the coordinates of a vector represent the correlation of the candidate term that the vector represents and the words in context-windows. As a result, the vectors' proximity can be employed to compare the distributional similarities of candidate terms. As suggested by Sahlgren (2006), the result is a geometric metaphor of meaning: a semantic space, which, following previous research such as Schütze (1993), can be named a *term-space model*.

[1] With the assumption that multi-token complex terms have no compositional semantics.

[2] This claim is not new. The *syntagmatic consequences* of hyponymy relationships in particular—and, the syntagmatic consequences of paradigmatic relationships in general—have been widely exploited in research literature. The aforementioned Hearst's (1992) patterns is, perhaps, the most familiar example. Hearst exploits the syntagmatic consequences of hyponym relationships to suggest patterns such as $\cdots X$ *and other* $Y \cdots$ for the automatic acquisition of hyponymy relationships. In this thesis, this linguistic phenomena is articulated in the framework of distributional semantic models in order to characterise co-hyponymy relationships.

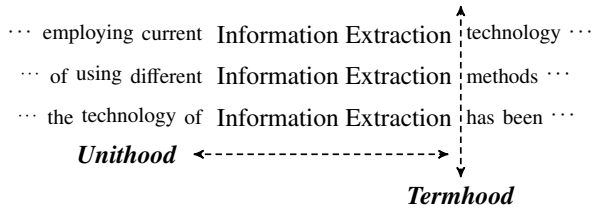


Figure 5.2: Unithood and termhood with respect to the terms usage in special corpus. In the given example, an ideal unithood measure identifies a strong association—that is, a syntagmatic relationship—between the two tokens *information* and *extraction*, and hence marks *information extraction* as a probable complex candidate term. Termhood, however, characterises the associations of specialised meanings to candidate terms: a paradigmatic relationship.

··· that arise when employing current **Information Extraction** technology to discover knowledge in ···
··· picture of the impact of using different **Information Extraction** methods for the offline construction of ···
··· on the development of the technology of **Information Extraction** has been stimulated by the Message ···

Figure 5.3: Illustration of a *context-window* of size 3 tokens that extend around a term: in the example above, this context-window is shown for the occurrences of the candidate term *information extraction* in three different sentences from a special corpus. For each occurrence of the candidate term *information extraction* in each line (i.e., a sentence), the context-window consists of words that are placed in rectangles. To construct a distributional model, the co-occurrences of *information extraction* and words within these context-windows are represented by a vector.

In this term-space model, a category of terms (i.e., co-hyponyms) is characterised using a set of *reference terms* (shown by R_s). R_s is a small number of terms that are manually annotated with their corresponding concept category. The distance between vectors that represent candidate terms and the vectors that represent R_s is assumed to determine the association of candidate terms to the concept categories represented by R_s. This association is computed using a k-nearest neighbours (k-nn) framework (Daelemans and van den Bosch, 2010). As is explained in Section 2.4 of Chapter 2, the memory-based k-nn learning technique provides a *similarity-based reasoning framework* that can be used to identify terms' categories without the need for formulating these associations using a meta-language, such as rules.

Previous research has confirmed the proposed term-space model's viability in capturing paradigmatic relationships between *words*, which can be taken as the evidence for the proposed method's practicability. However, as is described in Chapter 2, like other distributional approaches to semantics, finding a context-window's configurations that best characterises terms from similar concept categories is still a major research concern that must be investigated empirically. Besides the configuration of context-windows, the parameters of the classification framework are additional elements that influence the method's performance. The employed metric for similarity measurement, and the neighbourhood size (k) are the parameters that can be set differently in the k-nn algorithm. Understandably, a change in these parameters alters the observed results. To grasp the method's

behaviour, the effect of these parameters must be investigated empirically, too.

The remainder of this chapter is devoted to the delineation of the proposed method and the employed approach for its empirical investigation. Section 5.2 details the proposed method. The evaluation methodology and materials are described in Section 5.3. Subsequently, the observed results are reported in Section 5.4, which is followed by a summary in Section 5.7.

5.2 The Proposed Methodology

Figure 5.4 illustrates the method. It is assumed that an ATR system extracts a list of candidate terms and, perhaps, ranks them by its own weighting mechanism. The extracted list of candidate terms is then processed for constructing a vector space by scanning the corpus for the occurrences of the candidate terms. It is assumed that a small number of these candidate terms (e.g., 100) are annotated with their concept categories. Vectors that represent these annotated terms form a set of reference vectors R_s. In the constructed vector space, using a k-nn algorithm, R_s is employed to assign a concept category association weight c_w to the remaining candidate terms.

For a given candidate term represented by the vector $\vec{v}$, c_w is computed using

$$c_w(\vec{v}) = \sum_{i=1}^{k} s(\vec{v}, \vec{r}_i)\delta(\vec{r}_i), \tag{5.1}$$

where $s(\vec{v}, \vec{r})$ denotes similarity between $\vec{v}$ and $\vec{r} \in R_s$, in which R_s is sorted by $s(\vec{v}, \vec{r})$ in descending order. If $\vec{r}$ represents a term from the targeted category of concepts, then $\delta(\vec{r}) = 1$, otherwise $\delta(\vec{r}) = 0$. The function s can be defined in a number of ways; three widely used definitions are employed:[1]

- $s(\vec{v}, \vec{r}) = \cos(\vec{v}, \vec{r})$, that is, the cosine of the angles between $\vec{v}$ and $\vec{r}$;
- $s(\vec{v}, \vec{r}) = \frac{1}{1+\ell_2}$, where ℓ_2 is the Euclidean distance between $\vec{v}$ and $\vec{r}$; and
- $s(\vec{v}, \vec{r}) = \frac{1}{1+\ell_1}$, where ℓ_1 is the City block distance between $\vec{v}$ and $\vec{r}$.

As can be understood, the vector space construction is the major step in the proposed methodology, which is described in the following section.

5.2.1 Vector Space Construction Methodology

In distributional semantic models the curse of dimensionality is a common barrier, as is discussed in Chapter 2. In the proposed distributional method, due to the Zipfian distribution of terms and words in context-windows, the curse of dimensionality is an inevitable problem, too—that is, vectors that represent candidate terms are high-dimensional and sparse (i.e., most of the elements of vectors are zero). These properties of vectors hamper the subsequent classification process. To overcome this barrier, term-space models are

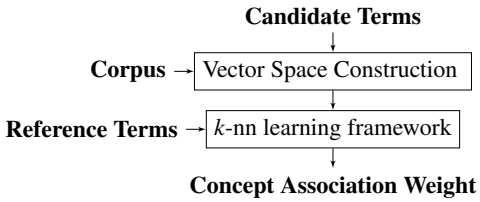


Figure 5.4: The proposed method for measuring the concept category associations.

constructed incrementally and at a reduced dimensionality using random projections techniques, which are proposed and justified in Chapter 4.

Each candidate term is assigned to an m-dimensional term vector $\vec{t}$. Term vectors are initially empty—that is, all the elements of $\vec{t}$ are set to zero. The corpus is then scanned for the occurrences of candidate terms and words that co-occur with them in context-windows. Each of these words is assigned exactly to one word vector $\vec{w}$. Similar to term vectors, word vectors are also m-dimensional. However, the elements w_j of each $\vec{w}$ are instantiated with random values with the following distributions:

$$w_j = \begin{cases} \lfloor \frac{-1}{U_1} \rfloor & \text{with probability } \frac{1}{2\alpha} \\ 0 & \text{with probability } 1 - \frac{1}{\alpha} \\ \lfloor \frac{1}{U_2} \rfloor & \text{with probability } \frac{1}{2\alpha} \end{cases}, \tag{5.2}$$

where α is a small value. As a result, most of the elements of w_j are set to zero and only a few have a non-zero value. Once a $\vec{w}$ is generated and assigned to a word, it is stored and kept for later usages.

If the similarity between $\vec{v}$ and $\vec{r}$ is measured using the cosine or the Euclidean distance (i.e., in an ℓ_2-normed space), then U_1 and U_2 are set to 1 and $\alpha = \mathrm{O}(\sqrt{|\vec{w}|})$, where $|\vec{w}|$ is the number of word vectors. In this case, $\vec{w}$ vectors resemble a random projection matrix that has a standard Gaussian distribution.[1] However, if the similarities are measured using the city block distance (i.e., in an ℓ_1-normed space), then U_1 and U_2 are two independent uniform random variables in $(0, 1)$ and $\alpha = \mathrm{O}(\sqrt{|\vec{w}|/100})$, where the constant factor 0.01 is an approximation of the sparsity of term-word co-occurrences in the corpus. In this case, $\vec{w}$ vectors resemble a random projection matrix with a standard Cauchy distribution.[2]

To capture the co-occurrence of a candidate term and a word in a context-window, the word vector $\vec{w}$ that represents the word is added to the term vector $\vec{v}$ that represents the candidate term—that is, $\vec{v} = \vec{t} + \vec{w}$. This procedure is repeated to capture all the co-occurrences of candidate terms and words that appear in context-windows in the input corpus. The result is a vector space that reflects the observed co-occurrences of terms and words, however, at the reduced dimension m.

Subsequent to the construction of a vector space using the method described above, the similarities between term vectors and reference term vectors in R_s must be computed. In

[1] See Section 2.3.4 of Chapter 2 for a long list of similarity measures.

[1] That is, a random projection matrix with asymptotic Gaussian distribution (see Chapter 4, Section 4.2 on Gaussian random projections).

[2] That is, a random projection matrix with asymptotic Cauchy distribution (see Chapter 4, Section 4.3 on random projections in ℓ_1-normed spaces).

Abstract	Sentence	Token	Type[1]
2,000	18,546	490,941	22,484 / 19,576

Table 5.1: A statistics summary of the GENIA corpus and its annotated terms.

the ℓ_2-normed constructed vector spaces, for the given vectors $\vec{v}$ and $\vec{u}$, the cosine between them is calculated using Equation 2.15 and their Euclidean distance using Equation 2.16. In the ℓ_1-normed spaces, the city block distance, however, is computed using the estimator proposed in Equation 4.21. Once computed, these distances and similarities between vectors are used to weight candidate terms according to Equation 5.1.

For instance, given the term *information extraction* in Figure 5.3, this term is first assigned to an empty m-dimensional term vector $\vec{t_{\text{ie}}}$. Assume that all the term's occurrences in the corpus are listed in this figure and the context-window is configured as shown (i.e., context-windows are stretched around the term for the size of three tokens). Then, each word placed in a rectangle is assigned exactly to one m-dimensional word vector $\vec{w}$. In this example, the result is 15 word vectors. The vector $\vec{t_{\text{ie}}}$ is then accumulated by these word vectors. Since the words *to* and *for* occur twice, their corresponding word vector is also accumulated twice. The generated word vectors are stored and used for constructing term vectors for candidate terms other than *information extraction*.

It becomes evident that the proposed method for constructing an ℓ_2-normed term-space model is equivalent to the random indexing technique. In the ℓ_1-normed spaces, however, the random Manhattan integer indexing (RMII) technique is employed. In Chapter 4, it is shown that the relative distances of vectors in these m-dimensional models are similar to the relative distances in the original high-dimensional vector spaces—that is, when a term-space model is constructed using the classic *one-dimension-per-context-element* method. Chapter 4 also discusses criteria for setting the parameters of the vector space construction—that is, m and α. Simply put, it is shown that the value of m is determined independently of the original dimension of the vector space (i.e., the number of distinct words that appear in context-windows). It is, however, determined by the number of term vectors $\vec{t}$ in the model. It is also described that α is decided by the original dimension and the sparseness of the vector space in its original dimension. These criteria are employed for setting the method's parameters in the reported evaluations.

5.3 The Evaluation Framework

5.3.1 Corpus and Performance Measure

To evaluate the plausibility of the proposed method and to determine its performance, a set of experiments over the *GENIA corpus* are carried out and the obtained results are

[1]The first row shows the number of distinct part-of-speech tagged tokens (normalised to lowercase) while the second row shows the number of distinct tokens irrespective of their assigned part-of-speech tag and when they are normalised to lowercase (which are used in the reported experiments).

$T_{Mention}$	$P_{Mention}$	$T_{Distinct}$	$P_{Distinct}$	$T_{Polysemy}$	$P_{Polysemy}$
92,722	37,660	34,077	8,900	1,373	403[1]

Table 5.2: Statistics of the employed terminological resource: terms and *protein terms* are respectively abbreviated by T and P (note P $\subset$ T). $T_{Polysemous}$ and $P_{Polysemous}$ show the number of distinct terms that are annotated with at least two different concept categories.

reported. The GENIA corpus is a collection of 2000 abstracts from the *domain of molecular biology* (Kim et al., 2003). The corpus comprises manual annotations of biological term mentions from several concept categories, which are organised in an ontology—also called the *GENIA ontology*. GENIA corpus is freely available and in the past decade has been used widely as a gold standard for benchmarking a variety of terminology mining methods. Table 5.1 gives a summary of the size of the corpus. Additional information about the GENIA corpus and its annotation process can be found in Kim et al. (2006).

In the GENIA ontology, terms are organised into 36 different categories such as *amino acids* (consisting of *proteins*, *peptides*, ...), *lipids*, *nucleic acids* (consisting of *DNA*, *RNA*, ...) and so on. To simplify the evaluation's reproducibility, a taxonomy of terms similar to the one suggested by Kim et al. (2004) in a shared-task for evaluating bio-entity taggers is employed. Manually annotated term mentions from the GENIA corpus[2] are thus collected to build a terminological resource, in which terms are organised according to the Kim et al.'s (2004) simplified taxonomy. To abridge the reports, unless otherwise stated, the focus is on the identification of terms belonging to the category of *proteins*—that is, the classification of *protein* and *non-protein* terms.

Table 5.2 shows the statistics for the extracted list of terms that is used as the gold standard. The reported statistics in Table 5.2 include mentions of both nested and non-nested terms. Amongst 97,876 '<cons>' mark-ups in the corpus that identify boundaries of terms, 5,154 mentions are not linked to the GENIA ontology and thus are not assigned to any concept category. From this list of 5,154 mentions with no concept category annotation, 1,440 distinct lexical units are not assigned to any concept category in the whole corpus. For instance, in

> ... are subject to tissue-specific and developmental stage-specific ...

the lexical unit *tissue-specific* is marked as a term but not assigned to any concept category. Similarly, in

> ... (SP and BP-14, 18, 19 kDs) isolated from splenic and brain cells...

splenic is marked as a term but not assigned to a category of concepts. These lexical units are removed from the list of compiled terms.

[2]Version 3.02, which can be downloaded from http://www.nactem.ac.uk/GENIA/current/GENIA-corpus/Term/GENIAcorpus3.02.tgz.

[1]In the GENIA corpus, protein terms themselves are classified into several categories such as *protein molecule*, *protein complex*, and so on. If this classification is considered, then the number of polysemous protein terms increases to 792.

	C	V	P
#Distinct Entry	58,558	19054	4,278
#Mentions	109,713	58,554	15,516
#Distinct$_{\text{Polysemous}}$	–	654	125
#Mentions$_{\text{Polysemous}}$	–	13,207	3,806

Table 5.3: Statistics of the extracted terminological resource using the YATEA system: candidate terms, valid terms and *protein terms* are respectively abbreviated by C, V, and P (note P $\subset$ V $\subset$ C). The statistics are also reported for Polysemous entries and their mentions in the corpus.

The collected mentions of terms are compiled independently of their concept category into a set of 34,077 distinct terms—that is, lexical units with identical surface structure are represented once in this set, even if they are annotated by two different concept categories. As reported in Table 5.2, only a small number of terms (i.e., 1,373) are polysemous and their mentions are annotated and classified in at least two concept categories. Amongst 8,900 terms that are classified as proteins, 403 terms are classified at least once in an additional concept category and as a result are considered polysemous (i.e., approximately 0.04% of all protein terms). For instance, in the following sentences

> ... using the murine B-cell lymphoma cell line A20, we show that ...
>
> ... correlate with expression of both BCL-2 and A20.

the mentions of the lexical unit *A20* are respectively annotated as a term of the concept categories *cell line* and *protein* (indicated by `G#cell_line` and `G#protein_molecule` in the GENIA corpus, respectively).

As stated earlier, the proposed method is built on top of an ATR system. Two methodologies are exploited for evaluations. In a set of experiments, in order to remove the effect of noise caused by the candidate term extraction process, the scope of ATR is limited only to the scoring and ranking process. Hence, it is assumed that the *noise-free* list of 34,077 terms in the GENIA corpus is known. Then, Frantzi et al.'s (2000) c-*value* score is employed to rank these terms by the frequencies that are obtained from the GENIA corpus.[1] This set of ranked terms is denoted by $\{\text{T}\}^{\text{c-}value}_{\text{ideal}}$. A random baseline for choosing a term from the category of proteins in $\{\text{T}\}^{\text{c-}value}_{\text{ideal}}$ thus approaches to $\frac{8900}{34077} = 0.261$.

The second set of experiments embraces errors caused by the candidate term extraction process. In order to get a ranked list of terms, sentences of part-of-speech tagged, lemmatised words from the GENIA corpus are fed to the YATEA system: a state-of-the-art term extraction system (Aubin and Hamon, 2006).[2] Using part-of-speech tag sequence patterns for the extraction of candidate terms and its internal scoring mechanism, YATEA pulls out a sorted set of 59,988 candidate terms from the GENIA corpus.[3] The

[1] The c-*value* score's definition is given by Equation 3.7, Chapter 3.

[2] Version 0.622, obtained from `http://search.cpan.org/~thhamon/Lingua-YaTeA/lib/Lingua/YaTeA.pm`.

[3] YATEA can be configured differently to boost its performance. For example, the part-of-speech sequence patterns for extracting candidate terms can be specified, or a set of verified terms may be provided to the system to enhance this process. However, to simplify reproducing the reported results, the system's default configuration is employed.

Denotation	Description
$\{T\}^{\text{c-}value}_{\text{ideal}}$	The set of terms extracted from the manual annotations in the GENIA corpus and sorted by the c-*value* score. This set does not contain invalid terms. The statistics for this set are reported in Table 5.2. Figure 5.5 shows the baseline performance computed in this set.
$\{T\}^{\text{YATEA}}_{\text{YATEA}}$	The set of candidate terms extracted and sorted by the YATEA system from the GENIA corpus. This set contains both valid and invalid terms. The statistics for this set are given in Table 5.3. Performance of protein term extraction in this set is reported in Figure 5.6.

Table 5.4: A summary of the resources that are employed in the experiments.

extracted terms are normalised by converting all their letters to lowercase; as a result, the size of the set is reduced to 58,558. This set of ranked terms is denoted by $\{T\}^{\text{YATEA}}_{\text{YATEA}}$. Amongst the set of 34,077 manually annotated terms derived as the gold standard form the GENIA corpus, 15,023 terms do not appear in $\{T\}^{\text{YATEA}}_{\text{YATEA}}$; 4,622 of these terms are from the concept category of proteins. As a result, $\{T\}^{\text{YATEA}}_{\text{YATEA}}$ contains only 4,278 terms that are once annotated as protein terms. Hence, a random baseline for choosing a term from the concept category of protein in $\{T\}^{\text{YATEA}}_{\text{YATEA}}$ approaches to $\frac{4278}{58558} = 0.073$. Table 5.3 provides a statistical summary of $\{T\}^{\text{YATEA}}_{\text{YATEA}}$. As can be inferred, errors caused by a candidate term extraction process can halve the recall in the extraction of a particular class of terms.

To measure the performance of the proposed method, two figures of merit are employed: *precision at n* ($P_{@n}$) and *non-interpolated precision at i* (NAP_i). $P_{@n}$ shows the proportion of protein terms in the set of top n candidate terms that are sorted in descending order by their assigned weights (i.e., c_w given by Equation 5.1). NAP_i, however, reports the average of precision for finding the first i protein terms in a set of sorted terms (see Chapter 3, page 96). For the baseline, I report $P_{@n}$ and NAP_i that are observed in $\{T\}^{\text{c-}value}_{\text{ideal}}$ and $\{T\}^{\text{YATEA}}_{\text{YATEA}}$, which are plotted in Figures 5.5 and 5.6. Table 5.4 gives a summary of the datasets and the obtained baselines employed for the evaluation.

5.3.2 Parameters for the Configuration of the Context-Window

In the proposed methodology, once candidate terms are extracted, they are represented as vectors. The incremental method explained in Section 5.2.1 is employed to collect and represent the co-occurrences of candidate terms and words in context-windows. The co-occurrences of candidate terms and words, however, can be collected from context-windows that are configured differently.

5.3.2.1 Direction

In the proposed distributional method, context-windows can be configured differently regarding the position of the candidate terms in them and the direction in which they are stretched. Context-windows can be expanded (a) to the *left side* of a candidate term to

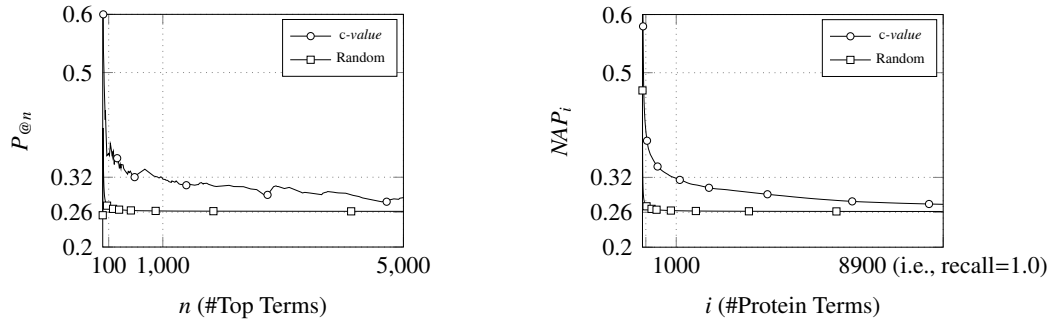


Figure 5.5: Baseline performance for protein term extraction in the $\{\mathrm{T}\}^{\textit{c-value}}_{\text{ideal}}$ ranked terms: (a) shows the proportion of protein terms in the top 5000 entries of the set of ranked candidate terms—that is, precision at n ($P_{@n}$) for $1 \leq n \leq 5000$; (b) shows the performance using non-interpolated precision at i (i.e., NAP_i) for $1 \leq i \leq 8900$; note that for $i = 8900$, recall is equal to 1.0. In both (a) and (b), a random baseline (computed by a simulation) is shown, too.

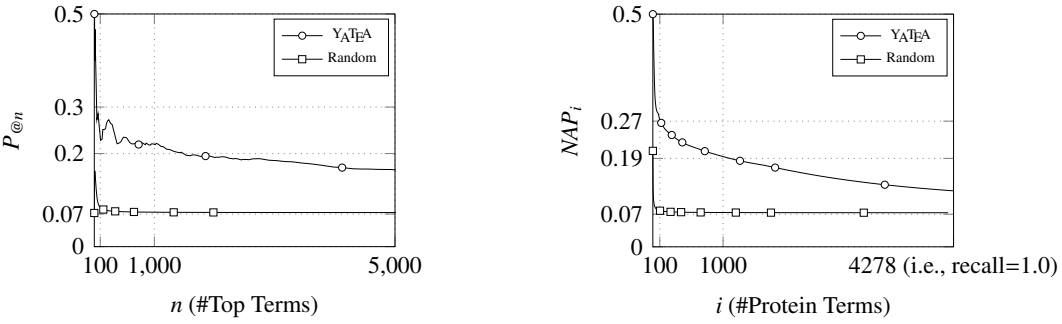


Figure 5.6: Baseline performance for protein term extraction in YATEA's extracted ranked terms (i.e., $\{\mathrm{T}\}^{\mathrm{YATEA}}_{\mathrm{YATEA}}$): (a) the proportion of protein terms in the top 5000 entries of the set of ranked candidate terms—that is, $P_{@n}$ for $1 \leq n \leq 5000$; (b) non-interpolated precision (i.e., NAP_i) for $1 \leq i \leq 4278$; note that for $i = 4278$, recall is equal to 1.0. In both (a) and (b), random baselines are also shown.

collect the co-occurrences of the candidate term with preceding words in each sentence of the corpus, (b) to the *right* side to collect co-occurrences with the succeeding words or (c) *around* the candidate term, that is, in both left and right directions. For instance, in Figure 5.3, words that are placed in rectangles show context-windows that expand around candidate terms.

5.3.2.2 Size

The size of context-windows can also be modified—that is, the extent of the region of either side of a term for collecting and counting its co-occurrences with neighbouring words. For instance, Figure 5.3 illustrates context-windows of size $t = 3$ tokens. As stated in the literature (e.g., see Lenci, 2008; Baroni et al., 2014), the optimum size of context-windows can only be established through experiments. However, research reports show that in contrast to wide context-windows (e.g., a paragraph or a document), narrow context-windows are more suitable to capture paradigmatic relations such as the intended concept category association in the proposed term classification task (e.g., Agirre et al., 2009; Zadeh and Handschuh, 2014b). In the performed experiments, therefore, the size of context-windows t is limited to $1 \leq t \leq 8$. The context-windows that expand around a candidate term are extended symmetrically in both directions.

5.3.2.3 Sequential order of words

Jones and Mewhort (2007) argue that the sequential order of words expresses information about lexical classes and grammatical behaviour, and therefore is important in the development of a comprehensive distributional semantic model. On the other hand, Landauer (2002) believes that 80% of the potential information in language is carried by the word choice regardless of the order in which they appear. He thus concludes that word order can be neglected in order to simplify the construction of vector spaces and their subsequent computations.[1] The influence of the inclusion of word order information on the performance of the method is investigated using a technique similar to the *permutation technique* proposed in Sahlgren et al. (2008); Recchia et al. (2015). However, instead of relying on intuition, I propose a mathematical justification based on the framework represented in Chapter 4.

5.3.2.3.1 Proposed method to capture the sequential order of words

One way to capture information about the sequential order of words in context-windows is to distinguish the appearances of words in different positions in these context-windows.[2]

This method could be best explained by giving the following example. In the first sentence of Figure 5.3, the word *technology* appears *after* the target term (i.e., *information extraction*) at the *position* $p = 1$ of the context-window. In the last sentence listed in

[1] Representing information about the order of words in context-windows usually entails appending additional dimensions to the underlying distributional model. Hence, computing similarities can demand more resources.

[2] Other methods are also conceivable, for instance, using n-grams, or even an additional vector space model that only captures the sequential order of words (e.g., as suggested by Jones and Mewhort, 2007).

Figure 5.3, the word *technology* also occurs, however, *before* the target term at the *position* $p = -2$. In this example, if the information about the sequential order of words is ignored, then the word *technology* is represented by only one standard basis $\vec{s}_t$ of the vector space—that is, one dimension of the model. The co-occurrence of the target term *information extraction* and the context word *technology* in these two sentences is then denoted by the coordinates $\vec{s}_t$ of the vector that represent the target term.

However, to capture information about the sequential order of words, the two appearances of the word *technology* must be distinguished and represented separately in the model. In doing so, additional dimensions must be appended to the model—one dimension per position per word. In the given example, this means that the occurrence of the word *technology* at the position $p = 1$ in context-windows must be presented by one standard basis $\vec{s}_a$ of the model, whereas the occurrence at the position $p = -2$ must be represented by another standard basis $\vec{s}_b$ of which $a \neq b$. In the same way, if the word *technology* appears at a location x other than $p = 1$ and -2 in context-windows, then it must be represented by an additional standard basis $\vec{s}_x$ of the vector space of which $x \neq a \neq b$. The co-occurrence of the target term *information extraction* and the context word *technology* at the two positions $p = 1$ and -2 is denoted by, respectively, the coordinates $\vec{s}_a$ and $\vec{s}_b$ of the vector that represents the target term *information extraction*.

If a vector space is constructed using the above-mentioned *one-dimension-per-context-element* methodology, then capturing information about the sequential order of words in context-windows drastically escalates the curse of dimensionality—as suggested by Landauer (2002), it is thus often discarded. However, as implied in Chapter 4, this problem can be easily obviated using random projections for the construction of a model. According to the principles discussed in Chapter 4 and based on the description given above, in order to capture the word order information in a vector space that is constructed using random projections, appearances of a word at different positions of context-windows are captured by assigning them to different word vectors.

Let us revisit the example given above and construct the model using random projections such as explained earlier in Section 5.2. If the word order information is ignored, then the word *technology* is assigned exactly to one word vector $\vec{w}_{\text{technology}}$, and both of its co-occurrences with the target term *information extraction* at $p = 1$ and -2 are captured by adding $\vec{w}_{\text{technology}}$ to the term vector $\vec{t}_{\text{information extraction}}$ that represents the target term at the reduced dimensionality—that is, $\vec{t}_{\text{information extraction}} = \vec{t}_{\text{information extraction}} + \vec{w}_{\text{technology}} + \vec{w}_{\text{technology}}$.

But, in order to model the sequential order of words, the appearances of the word *technology* at different positions in context-windows must be distinguished by assigning them to different word vectors. In the example above, the appearance of the word *technology* at $p = 1$ is captured by vector$\vec{w}^{p=1}_{\text{technology}}$ and its appearance at $p = -2$ is captured by $\vec{w}^{p=-2}_{\text{technology}}$ of which $\vec{w}^{p=-2}_{\text{technology}} \neq \vec{w}^{p=1}_{\text{technology}}$. The co-occurrences of the word *technology* and the term *information extraction* are then captured by accumulating these two different vectors to $\vec{t}_{\text{information extraction}}$—that is, $\vec{t}_{\text{information extraction}} = \vec{t}_{\text{information extraction}} + \vec{w}^{p=1}_{\text{technology}} + \vec{w}^{p=-2}_{\text{technology}}$. Both vectors $\vec{w}^{p=1}_{\text{technology}}$ and $\vec{w}^{p=-2}_{\text{technology}}$ are required to be stored for later usages—for example, to capture the co-occurrence of the word *technology* at $p = 1$ with another candidate term.

The method suggested above, however, is hampered by its required space for the stor-

age and retrieval of word vectors. Instead of creating several word vectors for representing appearances of one word at different positions of context-windows and storing them separately for later usages, one can use the *permutation technique*.

The main idea is that shuffling randomly created word vectors creates new random vectors that can be used to represent context words at various positions in context-windows. For example, this shuffling can be defined using a permutation function. This permutation function is defined using the location of context words in context-windows. In my implementation, a circular shift function serves as the permutation function (as suggested in Sahlgren et al., 2008, too). If p is the number of tokens before or after a candidate term and a word in a context-window (i.e., the position of the word in context-windows), then the word vector $\vec{w}$ that represents the word is shifted p times circularly to left or right prior to adding it to the candidate term's term vector. This circular shift of $\vec{w}$ results in a new random vector without the need to generate and store a new one. In this way, while the word order information is captured, the storage of additional word vectors is avoided. Hence, the method's space complexity enhances.[1]

5.3.3 Classification Parameters

In addition to various configurations of context-windows, the performance of the proposed term classification method is affected by the k-nn framework's parameters: (a) neighbourhood size selection (i.e., the value of k), (b) the size of the set of reference vectors (denoted by $|R_s|$)—that is, the number of training instances employed for the classification—and (c) the choice of similarity metric.

5.3.3.1 Neighbourhood size selection

The performance of k-nn is largely dependent on the value of k—that is, the neighbourhood size selection in the classification process. Using Bayesian mathematics, it is verified that if an infinite number of training samples are available (i.e., $|R_s| \to \infty$), then using a large value of k will result to the best-performing classification model (see Hastie et al., 2009, chap. 13). In the absence of a large R_s, in the employed memory-based learning framework, a small value for k can lead to *over-fitting* and sensitivity to noise, while a large neighbourhood estimation can reduce the discriminatory power of the classifier.

For a fixed R_s, if the underlying probability distribution of the term vectors in the vector space was known, the optimum k could be calculated. However, the underlying probability distribution is unknown and difficult to estimate. Therefore, the optimal value of k is usually obtained through experiments. To study the effect of neighbourhood size selection on the method's output, the performance is reported when k is set to different values. For instance, one can be interested in investigating whether the choice of k affects the choice for the best-performing context-windows configuration—that is to say, can one choose the most discriminative context-windows irrespective of the value of k?

[1] From an alternative perspective beyond the scope of this thesis, the suggested method is known as a *derandomisation* technique.

5.3.3.2 Similarity metrics

Last but not least, the choice of the method for similarity measurement between vectors—that is, $s(v, r_i)$ in Equation 5.1, thus its underlying metric—is another important factor that influences the method's performance. For instance, in a classification task similar to the proposed method, Weeds et al. (2005) suggest that the city block distance outperforms other similarity metrics such as the cosine measure. Therefore, the performance of classifiers that exploit different similarity and distance measures are reported. As implied in Section 5.2, the method's performance is assessed when using the Euclidean distance, the city block distance and the cosine similarity.

5.3.4 Setting the Parameters of Random Projection

For the experiments that are carried over the $\{\text{T}\}^{\text{c-}value}_{\text{ideal}}$ and $\{\text{T}\}^{\text{YATEA}}_{\text{YATEA}}$, vector spaces are constructed at the reduced dimensionality of $m = 2000$. Considering the small number of candidate terms in these datasets—that is, $34,077$ and $59,988$, respectively for $\{\text{T}\}^{\text{c-}value}_{\text{ideal}}$ and $\{\text{T}\}^{\text{YATEA}}_{\text{YATEA}}$—and based on the justification provided in Chapter 4, it can be verified that the dimensionality $m = 2000$ is large enough to construct models that preserve the relative pairwise distances between vectors in the original high-dimensional spaces.

For the construction of the ℓ_2-normed vector spaces at the reduced dimensionality of $m = 2000$, word vectors with 8 non-zero elements are employed. This means that in the reported experiments, the value of α from Equation 5.2 is set to 250. For the ℓ_1-normed vector space construction, however, word vectors with 40 non-zero elements are employed–that is, $\alpha = 50$. Considering the proposed approach for collecting co-occurrence frequencies, the original dimensionality of a vector space constructed in the employed datasets (shown by n) is a product of the size of the vocabulary in the GENIA corpus (i.e., 19,576, as reported in Table 5.1). Moreover, a model constructed at the original dimension is extremely sparse: depending on the configuration of context-windows, the sparseness of vectors (shown by β) in the reported experiments is $\beta < 10^{-3}$. Therefore, the suggested values for α are conservative choices that meet the criteria for the number of non-zero elements—that is, $\text{O}(n)$ and $\text{O}(\beta n)$ for the ℓ_2 and ℓ_1-normed spaces, respectively.

Considering the number of candidate terms that are represented in the constructed m-dimensional models and the original dimensionality of them (i.e., n), setting $m > 2000$ or using more non-zero elements in word vectors would not affect the obtained performances.

5.3.5 Evaluation Methodology

To find the best performing models, an exhaustive search is performed over the Cartesian product of a set of values for the parameters of (a) context-widows configuration (Section 5.3.2), and (b) the k-nn classification framework (Section 5.3.3).

In the reported empirical results, the proposed methodology in Section 5.2.1 is performed to construct several vector spaces from each combination of values that must be set for the configurations of context-windows. These vector spaces are constructed for the list of candidate terms in $\{\text{T}\}^{\text{c-}value}_{\text{ideal}}$ and $\{\text{T}\}^{\text{YATEA}}_{\text{YATEA}}$ (see Table 5.4), in which the co-occurrence

frequencies are collected from the GENIA corpus. Besides normalising text to lowercase letters and a *Penn Treebank* tokenisation, no other text preprocessing is performed. To summarise, context-windows are configured for:

- three directions: left, right, and around candidate terms;
- the size of context-windows is limited to t tokens, for $1 \leq t \leq 8$;
- inclusion and exclusion of information about sequential order of words.

Therefore, for each dataset 48 different vector spaces are constructed to encompass all the combinations of the values stated above.

The described term classification methodology in Section 5.2 is then employed to assign scores to the candidate terms in all the constructed vector spaces. The scoring procedure is also repeated for the combination of a set of values which can be set differently in the classification:

- three values for the neighbourhood size selection, that is $k = 1, 7, 25$;
- three similarity measures: cosine, the Euclidean, and the city block distance.

The top $n = 100$ entries from the list of ranked candidate terms in $\{\mathrm{T}\}^{\text{c-}value}_{\text{ideal}}$ and $\{\mathrm{T}\}^{\text{YATEA}}_{\text{YATEA}}$ are chosen to form the R_s. Hence, for each vector space, the scoring procedure is repeated 9 times in order to obtain 9 sets of ranked terms. The observed NAP_i in the obtained sets is then employed for their comparison and choosing the best combination of the evaluated parameters of the method.

5.4 Empirical Evaluations

5.4.1 Evaluation in $\{\mathrm{T}\}^{\text{c-}value}_{\text{ideal}}$: The Point of Departure

The first series of experiments are carried over $\{\mathrm{T}\}^{\text{c-}value}_{\text{ideal}}$ when the classification process is performed using a set of reference terms R_s of size 100 (i.e., $|R_s|$=100). In this experiment, R_s comprises of the top 100 entries from the ranked set of terms in $\{\mathrm{T}\}^{\text{c-}value}_{\text{ideal}}$ of which 36 entries are positive examples (i.e., protein terms). The experiments are duplicated for all of the context-window's configurations and the classification's parameters as explained in Section 5.3.5.

Tables 5.5, 5.6, and 5.7 report the results in detail when similarities are calculated using the cosine measure, the Euclidean distance, and the city block distance, respectively. The method's performance is denoted by the non-interpolated precision (i.e., NAP_i) for the identification of protein terms at $i = 200$ and $i = 8900$; note that $NAP_{i=200}$ and $NAP_{i=8900}$ denotes the method's performance when recall is 0.02 and 1.0, respectively. In these tables, the observed NAP_i over the list of sorted terms in $\{\mathrm{T}\}^{\text{c-}value}_{\text{ideal}}$ is reported as a baseline (see Figure 5.5). Figures 5.7 and 5.8 summarise and plot the reported numbers in these tables.

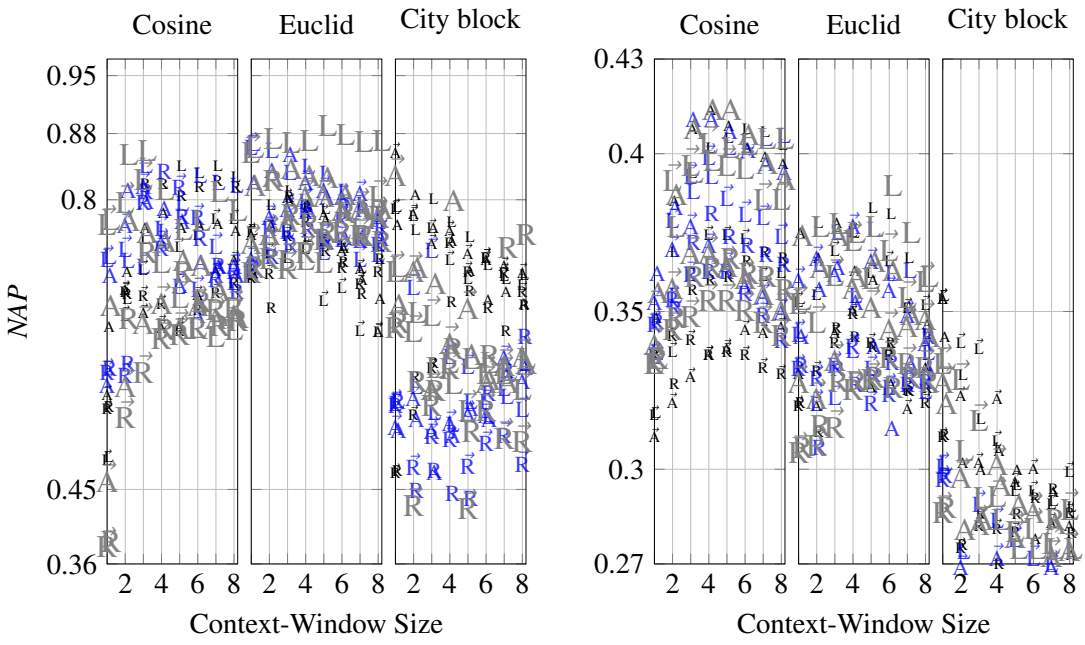
Context		$NAP_{i=200}$			$NAP_{i=8900}$		
dir	size	k			k		
		1	7	25	1	5	25
Around	1	0.647	0.709	0.673	0.348	0.362	0.356
	2	0.713	0.811	0.775	0.385	0.383	0.387
	3	0.766	0.798	0.761	0.408	0.411	0.408
	4	0.784	0.79	0.75	0.414	0.411	0.414
	5	0.766	0.774	0.727	0.409	0.407	0.414
	6	0.769	0.725	0.721	0.402	0.401	0.406
	7	0.771	0.72	0.691	0.399	0.397	0.397
	8	0.763	0.709	0.658	0.397	0.394	0.392
Left	1	0.489	0.732	0.776	0.318	0.337	0.335
	2	0.68	0.727	0.855	0.369	0.355	0.377
	3	0.753	0.842	0.86	0.394	0.388	0.394
	4	0.82	0.762	0.813	0.402	0.393	0.398
	5	0.841	0.796	0.764	0.405	0.402	0.402
	6	0.833	0.83	0.775	0.408	0.404	0.403
	7	0.841	0.821	0.81	0.407	0.405	0.405
	8	0.828	0.817	0.787	0.403	0.406	0.405
Right	1	0.55	0.596	0.387	0.337	0.347	0.335
	2	0.686	0.602	0.657	0.327	0.351	0.348
	3	0.819	0.81	0.736	0.357	0.372	0.361
	4	0.82	0.834	0.643	0.36	0.381	0.364
	5	0.805	0.78	0.714	0.361	0.374	0.367
	6	0.816	0.753	0.677	0.366	0.371	0.362
	7	0.823	0.721	0.648	0.369	0.368	0.361
	8	0.816	0.69	0.654	0.367	0.365	0.357
Baseline		**0.364**			**0.273**		

(a) Sequential Order of Words Discarded

$NAP_{i=200}$			$NAP_{i=8900}$		
k			k		
1	7	25	1	5	25
0.568	0.58	0.466	0.311	0.354	0.341
0.694	0.773	0.58	0.322	0.371	0.359
0.666	0.821	0.657	0.33	0.378	0.37
0.648	0.752	0.681	0.337	0.372	0.37
0.65	0.701	0.65	0.339	0.366	0.366
0.688	0.665	0.678	0.345	0.359	0.362
0.717	0.713	0.663	0.344	0.356	0.357
0.73	0.724	0.666	0.346	0.351	0.352
0.489	0.732	0.776	0.318	0.337	0.335
0.663	0.744	0.803	0.338	0.38	0.393
0.719	0.724	0.831	0.362	0.394	0.398
0.653	0.77	0.73	0.375	0.4	0.405
0.746	0.7	0.688	0.375	0.387	0.398
0.761	0.696	0.652	0.374	0.382	0.397
0.78	0.748	0.641	0.362	0.381	0.387
0.78	0.705	0.649	0.363	0.375	0.387
0.55	0.596	0.387	0.337	0.347	0.335
0.692	0.592	0.543	0.343	0.355	0.346
0.687	0.801	0.598	0.342	0.363	0.35
0.677	0.746	0.638	0.337	0.364	0.355
0.645	0.819	0.643	0.338	0.364	0.355
0.67	0.783	0.651	0.337	0.357	0.352
0.691	0.708	0.673	0.333	0.35	0.348
0.703	0.718	0.676	0.331	0.342	0.344
0.364			**0.273**		

(b) Sequential Order of Words Encoded

Table 5.5: The performances observed over the $\{T\}^{\text{c-}value}_{\text{ideal}}$ when $|R_s| = 100$ and similarities are computed using the *cosine* between vectors. The performance is shown with regards to the observed NAP_i, for $i = 200$ (i.e., recall = 0.02) and $i = 8900$ (i.e., recall = 1.0). The baseline shows the computed *NAP* when terms are sorted using the c-*value* (see Figure 5.5); (a) denotes the performance of models that ignore the sequential order of words in context-windows, whereas (b) shows the performance when this information is encoded.



(a) $\boldsymbol{NAP_{i=200}}$ (b) $\boldsymbol{NAP_{i=8900}}$

Figure 5.7: The results obtained over $\{\mathrm{T}\}^{\text{c-value}}_{\text{ideal}}$ when $|R_s| = 100$: The y-axis shows the observed performances (i.e., NAP_i) for the identification of protein terms when the method's parameters are set differently. Each box in the sub-figures denotes the performance for each of the employed similarity metric. In these boxes, the x-axis shows the size of context-windows. The letters A, L, and R denote the direction in which context-windows are stretched (i.e., respectively, Around, Left, or the Right side of the candidate terms). Models that encode word order information are denoted using the $\vec{\square}$ on top of the letters. The size of letters, however, shows the value of k. The smallest size denotes $k = 1$ (black colour), while the largest size denotes $k = 25$ (grey colour); the medium size represents $k = 7$ (blue colour). In these plots, the minimum value of y-axis shows the baseline.

First and foremost, a glance at Figures 5.7a and 5.7b, and Figures 5.8a and 5.8b, indicates that choosing the best performing configuration for the method's parameters is subject to the chosen i for computing and reporting NAP_i as the performance measure. Besides this observation, these figures suggest that obtaining the best performance is drastically dependant on the choice of similarity metrics (e.g., by comparing the average of the obtained performances over parameters of the method). If the performance is measured using NAP_i at a small i such as 200 in this experiment—that is, if the intention is the extraction of a small number of terms and hence precision is more important than recall—then the Euclidean distance seems to be a more desirable choice than the city block distance or the cosine measure. However, for NAP_i for a large i, for example $i = 8900$ (i.e., if a high recall is intended), then the cosine similarity seems to be a more robust choice than the other evaluated similarity metrics.

When the method's performance is evaluated using $NAP_{i=200}$, then the Euclidean dis-

Context		$NAP_{i=200}$			$NAP_{i=8900}$		
dir	size	k			k		
		1	7	25	1	5	25
Around	1	0.712	0.821	0.817	0.375	0.342	0.374
	2	0.769	0.836	0.828	0.367	0.357	0.364
	3	0.799	0.855	0.838	0.378	0.364	0.374
	4	0.794	0.788	0.829	0.363	0.353	0.361
	5	0.776	0.803	0.827	0.36	0.339	0.366
	6	0.742	0.755	0.806	0.327	0.313	0.363
	7	0.704	0.756	0.788	0.352	0.334	0.349
	8	0.68	0.753	0.79	0.346	0.332	0.342
Left	1	0.764	0.868	0.863	0.354	0.344	0.354
	2	0.794	0.86	0.874	0.378	0.365	0.377
	3	0.811	0.833	0.869	0.377	0.375	0.378
	4	0.756	0.84	0.872	0.375	0.355	0.375
	5	0.789	0.832	0.89	0.383	0.376	0.377
	6	0.744	0.81	0.879	0.381	0.362	0.39
	7	0.718	0.794	0.871	0.367	0.363	0.375
	8	0.745	0.783	0.872	0.343	0.337	0.361
Right	1	0.709	0.718	0.718	0.321	0.333	0.307
	2	0.67	0.732	0.823	0.312	0.307	0.314
	3	0.804	0.746	0.782	0.345	0.332	0.328
	4	0.792	0.764	0.772	0.342	0.327	0.328
	5	0.738	0.75	0.761	0.339	0.33	0.329
	6	0.725	0.792	0.785	0.34	0.334	0.326
	7	0.687	0.788	0.757	0.325	0.329	0.329
	8	0.729	0.767	0.75	0.326	0.342	0.328
Baseline		**0.364**			**0.273**		

(a) Sequential Order of Words Discarded

$NAP_{i=200}$			$NAP_{i=8900}$		
k			k		
1	7	25	1	5	25
0.761	0.732	0.753	0.331	0.361	0.329
0.768	0.783	0.764	0.321	0.368	0.324
0.805	0.793	0.78	0.344	0.363	0.336
0.789	0.825	0.81	0.351	0.378	0.359
0.785	0.816	0.81	0.327	0.367	0.353
0.741	0.807	0.793	0.339	0.358	0.346
0.72	0.812	0.796	0.32	0.35	0.347
0.643	0.778	0.785	0.333	0.341	0.349
0.764	0.868	0.863	0.354	0.344	0.354
0.739	0.752	0.751	0.356	0.322	0.348
0.733	0.769	0.733	0.37	0.324	0.363
0.744	0.802	0.734	0.376	0.339	0.374
0.681	0.741	0.73	0.363	0.333	0.362
0.697	0.757	0.77	0.367	0.327	0.371
0.645	0.733	0.771	0.352	0.329	0.356
0.644	0.738	0.81	0.353	0.326	0.361
0.709	0.718	0.718	0.321	0.333	0.307
0.714	0.77	0.735	0.332	0.33	0.309
0.76	0.766	0.723	0.341	0.334	0.315
0.777	0.79	0.742	0.35	0.34	0.327
0.752	0.76	0.758	0.341	0.322	0.33
0.721	0.781	0.767	0.337	0.325	0.337
0.702	0.756	0.763	0.326	0.328	0.335
0.716	0.751	0.786	0.322	0.329	0.334
0.364			**0.273**		

(b) Sequential Order of Words Encoded

Table 5.6: The results observed in $\{\mathrm{T}\}^{\text{c-}value}_{\text{ideal}}$ when $|R_s| = 100$ and similarities are computed using the *Euclidean* distance. The presentation format is similar to Table 5.5.

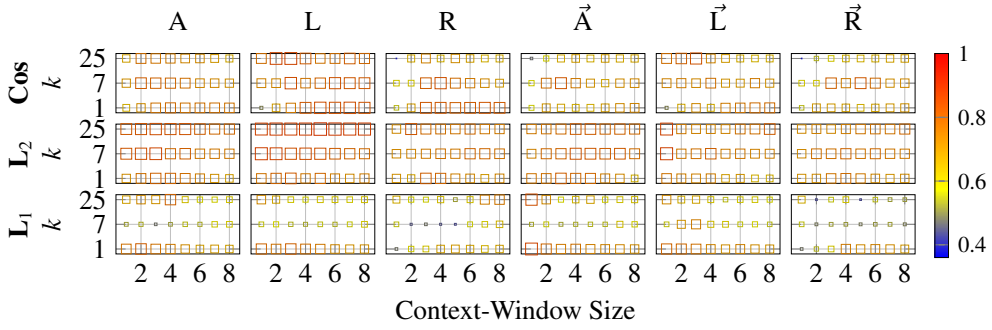
Context		$NAP_{i=200}$			$NAP_{i=8900}$		
dir	size	k			k		
		1	7	25	1	5	25
Around	1	0.77	0.521	0.674	0.342	0.259	0.321
	2	0.806	0.535	0.718	0.302	0.259	0.281
	3	0.766	0.471	0.699	0.3	0.253	0.284
	4	0.751	0.516	0.799	0.306	0.262	0.295
	5	0.714	0.548	0.613	0.3	0.262	0.289
	6	0.679	0.535	0.583	0.278	0.263	0.269
	7	0.689	0.589	0.58	0.293	0.264	0.283
	8	0.675	0.598	0.602	0.273	0.261	0.266
Left	1	0.792	0.558	0.72	0.355	0.302	0.335
	2	0.764	0.575	0.664	0.33	0.274	0.306
	3	0.801	0.543	0.597	0.316	0.264	0.291
	4	0.773	0.529	0.578	0.307	0.269	0.284
	5	0.755	0.556	0.608	0.295	0.266	0.275
	6	0.721	0.538	0.603	0.292	0.266	0.275
	7	0.705	0.563	0.594	0.29	0.265	0.275
	8	0.709	0.602	0.608	0.288	0.267	0.275
Right	1	0.473	0.556	0.652	0.311	0.298	0.288
	2	0.586	0.449	0.581	0.276	0.243	0.25
	3	0.603	0.473	0.57	0.286	0.251	0.257
	4	0.677	0.446	0.621	0.27	0.242	0.25
	5	0.69	0.439	0.52	0.286	0.264	0.249
	6	0.67	0.566	0.612	0.291	0.265	0.267
	7	0.642	0.594	0.743	0.294	0.257	0.284
	8	0.673	0.635	0.758	0.28	0.256	0.267
Baseline		**0.364**			**0.273**		

(a) Sequential Order of Words Discarded

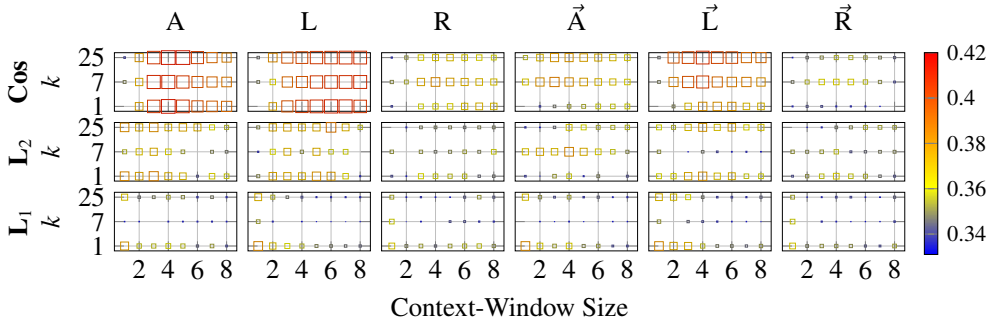
$NAP_{i=200}$			$NAP_{i=8900}$		
k			k		
1	7	25	1	5	25
0.86	0.529	0.836	0.355	0.265	0.329
0.786	0.564	0.719	0.324	0.27	0.299
0.75	0.525	0.577	0.303	0.267	0.284
0.766	0.534	0.633	0.323	0.273	0.303
0.738	0.545	0.675	0.297	0.267	0.28
0.73	0.544	0.585	0.301	0.266	0.288
0.723	0.614	0.606	0.284	0.27	0.274
0.715	0.585	0.617	0.283	0.269	0.276
0.792	0.558	0.72	0.355	0.302	0.335
0.784	0.699	0.647	0.341	0.266	0.33
0.769	0.743	0.666	0.339	0.29	0.317
0.73	0.615	0.609	0.31	0.285	0.293
0.708	0.603	0.552	0.294	0.279	0.282
0.733	0.561	0.596	0.294	0.273	0.283
0.711	0.576	0.596	0.29	0.271	0.278
0.7	0.551	0.577	0.3	0.268	0.289
0.473	0.556	0.652	0.311	0.298	0.288
0.543	0.481	0.437	0.276	0.243	0.247
0.6	0.519	0.558	0.283	0.257	0.254
0.751	0.515	0.602	0.282	0.252	0.263
0.716	0.482	0.433	0.281	0.246	0.263
0.733	0.51	0.551	0.268	0.247	0.251
0.715	0.52	0.518	0.265	0.247	0.251
0.692	0.485	0.514	0.287	0.261	0.269
0.364			**0.273**		

(b) Sequential Order of Words Encoded

Table 5.7: The performances observed in $\{\mathrm{T}\}^{\text{c-}value}_{\text{ideal}}$ when $|R_s| = 100$ and similarities are computed using the *city block* distance. The presentation format is similar to Table 5.5.



(a) $NAP_{i=200}$

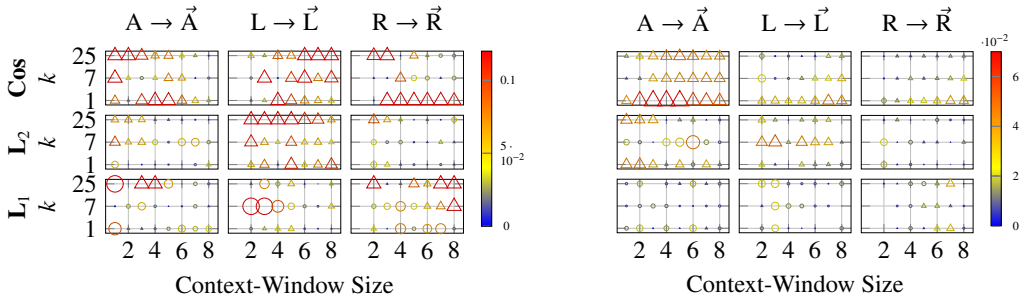


(b) $NAP_{i=8900}$

Figure 5.8: The results obtained over $\{T\}^{c\text{-}value}_{ideal}$ when $|R_s| = 100$: shown are the obtained results grouped by the type of similarity metrics and values of k (i.e., the rows), as well as the direction and the size of context-windows (i.e., the columns). The letters A, L, and R denote the direction in which context-windows are stretched (i.e., respectively, Around, Left, or the Right side of the candidate terms). Models that encode word order information are denoted using the $\vec{\square}$ on top of the letters. The size of squares in these plots denote the value for NAP_i at $i = 200$ (i.e., Figure 5.8a), and $i = 8900$ (i.e, Figure 5.8b).

tance seems to be the least sensitive similarity metric to the changes in the values of the remaining method's parameters—that is, it shows the least variance in the performance. Although this behaviour of the Euclidean distance changes when the performance is measured using $NAP_{i=8900}$. When it comes to choosing a value for the neighbourhood size in the classification process (i.e., k), the city block distance is the most sensitive measure. Except $k = 1$, the city block distance does not show an acceptable performance in these experiments.

In these experiments, context-windows that extend to the left side or around the terms usually outperform context-windows that only extend to the right of candidate terms. If the obtained performances are averaged independently of the value of k, the employed similarity metric, and the size of context-windows, then context-windows that extend around terms are preferable to those that only extend to the left side of candidate terms. As can be inferred from the reported results, encoding information about the sequential



(a) $NAP_{i=200}$ (b) $NAP_{i=8900}$

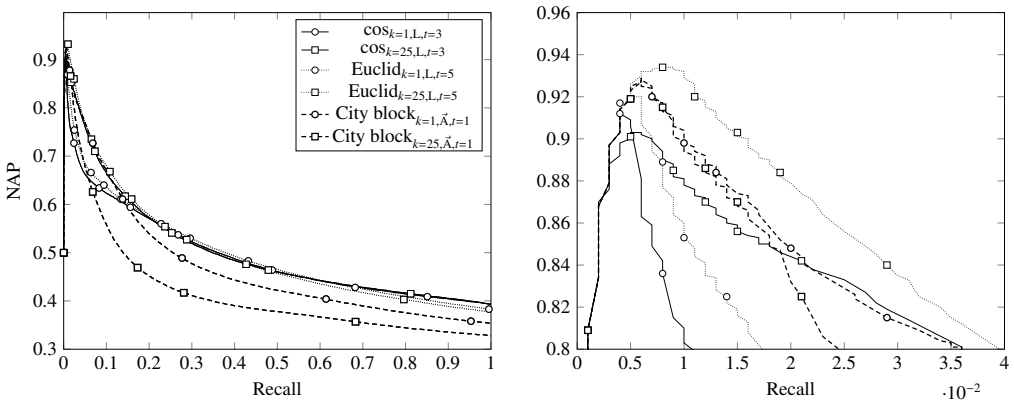
Figure 5.9: Observed differences in the performance of method when the sequential order of words are encoded in the context-windows. The letters A, L, and R denote the three directions of Around, Left, or Right side of the candidate terms, respectively. Accordingly, the notation X $\rightarrow \vec{\text{X}}$ shows the difference in the performance of the models constructed by collecting co-occurrences at the direction X before and after encoding information about the sequential order of words. An increase in the performance is marked by a circle, whereas a reduction is shown by a triangle. The size of shapes shows the intensity.

order of words in context-windows does not necessarily enhance the observed result. The effect of encoding this information in the observed performances is shown in detail in Figure 5.9. As shown in the figure, except when using the city block distance for measuring similarities, encoding word order information has a negative impact on the performance.

Choosing the best size for context-windows is also largely dependant on the chosen similarity metric and the direction in which context-windows are extended. However, according to the obtained results, context-windows of size $3 \leq t \leq 6$ tokens are often amongst the top performers. As shown in Figure 5.7, the city block distance is again an exception in which a small size for context-windows—that is, $1 \leq t \leq 2$ tokens—outperforms larger sizes of context-windows. Although the cosine measure on average results in a higher performance (particularly at 100% recall), using a distance metric is preferable to the use of cosine similarity if a small recall is targeted (see Figure 5.10).

5.4.1.1 Using an entity tagger as an additional baseline

To have a better understanding of the reported performance measures, an additional baseline is introduced. The same set of annotated candidate terms (i.e., R_s) used in the experiment reported in Section 5.4.1 is employed to train a biomedical named entity recogniser. Namely, the *ABNER* system, an entity tagger based on conditional random fields, is employed. ABNER exploits a variety of orthographic and contextual features designed for the analysis of biology text (Settles, 2005). If ABNER is trained using all the manual annotation that are provided for the mentions of terms in the GENIA corpus, it achieves a recall of 77.8 and precision of 68.1 for extracting protein terms. Consequently, it is one of the top-performing bio-entity recognition systems for extracting protein terms (see Kim et al., 2004, for the performance comparison of ABNER and several other entity taggers in a shared task).



(a) (b)

Figure 5.10: The performance of similarity metrics across the complete range of recall values: (a) shows NAP_i for $0 < i \leq 8900$ (i.e., the complete range of values for recall) for the three employed similarity metrics for particular configurations of context-windows. For recall less than 2%, as shown with minute details in (b), the choice of a distance measure results in a better performance than the use of cosine.

To compute the second baseline, ABNER is trained using the mentions of the protein term that appear in R_s. To ensure that the provided training dataset for the development of the tagger's model is noise-free, manual annotations in the GENIA corpus are used to mark each of the term mentions. The 36 protein terms in R_s are mentioned 1,321 times in the GENIA corpus. Therefore, it is worthwhile mentioning that although the entity tagger is developed by the same number of distinct terms as appeared in R_s, in practice, preparing the training dataset for the development of the entity tagger's model requires more manual effort than preparing R_s.

After training the ABNER tagger, the obtained model is reapplied to the same corpus (i.e., GENIA) in order to extract additional mentions of protein terms. The extracted terms using ABNER are collected in a set and the number of distinct valid and invalid protein terms are reported as the baseline. The resulting model can only identify an additional 16 protein terms out of the remaining 8,864 terms in the corpus. Simply put, and as suggested in the introduction of this chapter, the 1,321 mentions of the 36 protein terms in R_s are not sufficient for the training of the ABNER system and extracting additional terms from the concept category of proteins.

5.4.2 Evaluation of the Method in $\{T\}^{Y_A T_E A}_{Y_A T_E A}$: The Method's Performance in the Presence of Noise

In this section, the observed results are reported when the method is applied to $\{T\}^{Y_A T_E A}_{Y_A T_E A}$. As stated earlier, 67% of candidate terms in $\{T\}^{Y_A T_E A}_{Y_A T_E A}$ are *invalid* terms (see Table 5.3). Hence, the performed experiments let us study the effect of noise caused by the process of candidate term extraction in the method's performance. Therefore, the evaluation process described in the previous section is repeated over the ranked set of candidate terms in

Context		$NAP_{i=98}$			$NAP_{i=4278}$		
dir	size	k			k		
		1	7	25	1	7	25
Around	1	0.55(0.77)	0.31(0.81)	0.22(0.69)	0.11(0.36)	0.13(0.42)	0.14(0.42)
	2	0.62(0.79)	0.49(0.8)	0.47(0.75)	0.13(0.36)	0.15(0.41)	0.16(0.42)
	3	0.68(0.85)	0.47(0.73)	0.66(0.77)	0.14(0.37)	0.15(0.39)	0.17(0.4)
	4	0.68(0.85)	0.39(0.69)	0.53(0.73)	0.14(0.37)	0.15(0.38)	0.16(0.39)
	5	0.69(0.86)	0.48(0.8)	0.54(0.8)	0.15(0.37)	0.15(0.38)	0.16(0.38)
	6	0.68(0.88)	0.37(0.71)	0.41(0.61)	0.14(0.37)	0.14(0.37)	0.15(0.37)
	7	0.71(0.87)	0.41(0.71)	0.39(0.59)	0.14(0.37)	0.14(0.37)	0.14(0.37)
	8	0.71(0.87)	0.37(0.7)	0.37(0.57)	0.14(0.37)	0.13(0.36)	0.14(0.37)
Left	1	0.36(0.5)	0.24(0.67)	0.41(0.8)	0.1(0.39)	0.12(0.43)	0.13(0.46)
	2	0.6(0.78)	0.39(0.73)	0.46(0.8)	0.13(0.4)	0.14(0.43)	0.16(0.45)
	3	0.69(0.85)	0.49(0.71)	0.48(0.77)	0.13(0.38)	0.15(0.42)	0.16(0.43)
	4	0.72(0.85)	0.49(0.71)	0.56(0.79)	0.13(0.37)	0.15(0.41)	0.17(0.43)
	5	0.74(0.86)	0.52(0.72)	0.53(0.8)	0.14(0.37)	0.15(0.4)	0.17(0.42)
	6	0.76(0.89)	0.47(0.63)	0.47(0.69)	0.14(0.37)	0.15(0.4)	0.16(0.41)
	7	0.78(0.91)	0.41(0.62)	0.44(0.71)	0.14(0.37)	0.14(0.39)	0.16(0.41)
	8	0.78(0.91)	0.41(0.63)	0.43(0.67)	0.15(0.38)	0.14(0.39)	0.15(0.4)
Right	1	0.36(0.41)	0.22(0.41)	0.16(0.37)	0.09(0.31)	0.1(0.33)	0.09(0.33)
	2	0.42(0.43)	0.28(0.41)	0.41(0.58)	0.09(0.32)	0.1(0.32)	0.1(0.33)
	3	0.45(0.46)	0.33(0.52)	0.38(0.59)	0.09(0.32)	0.1(0.33)	0.11(0.33)
	4	0.49(0.5)	0.37(0.58)	0.53(0.69)	0.1(0.33)	0.11(0.33)	0.12(0.34)
	5	0.55(0.6)	0.35(0.57)	0.43(0.71)	0.1(0.33)	0.11(0.33)	0.12(0.35)
	6	0.55(0.68)	0.28(0.54)	0.37(0.63)	0.1(0.34)	0.1(0.33)	0.11(0.34)
	7	0.57(0.69)	0.29(0.51)	0.34(0.58)	0.11(0.34)	0.11(0.33)	0.11(0.34)
	8	0.59(0.74)	0.32(0.56)	0.3(0.54)	0.11(0.34)	0.11(0.34)	0.11(0.34)
Baseline		**0.273(0.87)**			**0.12(0.5)**		

(a) Sequential Order of Words Discarded

$NAP_{i=98}$			$NAP_{i=4278}$		
k			k		
1	7	25	1	7	25
0.42(0.56)	0.25(0.52)	0.28(0.56)	0.09(0.34)	0.11(0.37)	0.11(0.36)
0.47(0.6)	0.33(0.55)	0.29(0.46)	0.09(0.34)	0.11(0.37)	0.12(0.36)
0.43(0.58)	0.29(0.49)	0.38(0.53)	0.09(0.33)	0.11(0.35)	0.12(0.34)
0.45(0.6)	0.3(0.51)	0.33(0.5)	0.09(0.33)	0.1(0.34)	0.11(0.34)
0.46(0.63)	0.24(0.47)	0.24(0.41)	0.09(0.33)	0.1(0.33)	0.11(0.33)
0.48(0.64)	0.27(0.45)	0.32(0.45)	0.09(0.34)	0.1(0.34)	0.11(0.34)
0.5(0.66)	0.2(0.38)	0.3(0.43)	0.09(0.33)	0.1(0.33)	0.1(0.33)
0.53(0.65)	0.19(0.37)	0.29(0.44)	0.09(0.33)	0.09(0.33)	0.1(0.33)
0.36(0.5)	0.24(0.67)	0.41(0.8)	0.1(0.39)	0.12(0.43)	0.13(0.46)
0.44(0.5)	0.39(0.61)	0.52(0.65)	0.09(0.32)	0.11(0.37)	0.12(0.37)
0.52(0.68)	0.38(0.47)	0.36(0.5)	0.09(0.34)	0.1(0.35)	0.11(0.35)
0.6(0.72)	0.37(0.5)	0.34(0.59)	0.1(0.34)	0.11(0.36)	0.11(0.35)
0.6(0.73)	0.33(0.51)	0.39(0.59)	0.11(0.34)	0.11(0.36)	0.11(0.35)
0.63(0.77)	0.37(0.61)	0.47(0.77)	0.1(0.34)	0.12(0.36)	0.14(0.4)
0.63(0.79)	0.33(0.56)	0.5(0.71)	0.1(0.34)	0.11(0.36)	0.13(0.38)
0.62(0.79)	0.41(0.58)	0.49(0.74)	0.1(0.34)	0.11(0.35)	0.13(0.38)
0.36(0.41)	0.22(0.41)	0.16(0.37)	0.09(0.31)	0.1(0.33)	0.09(0.33)
0.5(0.6)	0.34(0.56)	0.21(0.42)	0.1(0.32)	0.1(0.33)	0.1(0.33)
0.53(0.59)	0.33(0.51)	0.25(0.43)	0.09(0.32)	0.1(0.32)	0.1(0.33)
0.53(0.65)	0.26(0.49)	0.25(0.43)	0.09(0.32)	0.1(0.32)	0.1(0.32)
0.49(0.61)	0.28(0.49)	0.23(0.39)	0.09(0.32)	0.1(0.32)	0.1(0.32)
0.5(0.62)	0.21(0.44)	0.24(0.42)	0.09(0.32)	0.1(0.32)	0.1(0.32)
0.48(0.6)	0.2(0.42)	0.24(0.36)	0.09(0.32)	0.09(0.31)	0.1(0.32)
0.54(0.66)	0.21(0.43)	0.24(0.37)	0.09(0.32)	0.09(0.31)	0.1(0.32)
0.273(0.87)			**0.12(0.5)**		

(b) Sequential Order of Words Encoded

Table 5.8: The performances obtained over the $\{\mathrm{T}\}^{\mathrm{YATEA}}_{\mathrm{YATEA}}$ when $|R_s| = 100$ and similarities are computed using the *cosine* between vectors. The performance is shown with regards to the observed *NAP*, for $i = 98$ (i.e., recall = 0.02) and $i = 4278$ (i.e., recall = 1.0). Parenthesised numbers show *NAP* for valid terms. The baseline shows the computed *NAP* when terms are sorted using the YATEA's weighting mechanism (see Figure 5.6); (a) shows the performance of models that ignore the sequential order of words in context-windows, whereas (b) shows the performance when this information is encoded.

$\{\mathrm{T}\}^{\mathrm{YATEA}}_{\mathrm{YATEA}}$ using $|R_s| = 100$ (similar to the previous experiment). As a result, R_s contains 22 protein terms (i.e., positive examples). Tables 5.8, 5.9, and 5.10 report the performance using NAP_i at $i = 98$ (i.e., recall =0.02) and $i = 4278$ (i.e., recall =1) when similarities are calculated using the cosine measure, the Euclidean distance, and the city block distance, respectively. Apart from the method's performance for the identification of protein terms, these tables report NAP_i when the goal is to extract valid terms.

In this set of experiments, the cosine similarity outperforms both the Euclidean and the city block distance with a large margin. Although the method shows an acceptable performance for a small recall such as 0.02% (i.e., $NAP_{i=98}$ as shown in Figure 5.11a), its performance drastically drops for a complete recall (i.e., $NAP_{i=4278}$). If the goal is to extract all the protein terms in the corpus, the method underperforms the ATR's ranking baseline and it shows a performance similar to the random baseline (Figure 5.11b). As reported in the tables, the obtained scores for sorting candidate terms using the proposed method also decreases the number of valid terms in the obtained ranked sets of terms.

The major sources of errors in this set of experiments are invalid candidate terms in which a valid protein term appears nested. For example, in the performed experiments, *I kappa B* is a protein term, which often appears at the top of the obtained ranked lists.

Context		$NAP_{i=98}$			$NAP_{i=4278}$		
dir	size	k			k		
		1	7	25	1	7	25
Around	1	0.29(0.18)	0.38(0.44)	0.59(0.95)	0.08(0.3)	0.08(0.31)	0.12(0.5)
	2	0.27(0.17)	0.29(0.51)	0.6(0.9)	0.07(0.3)	0.08(0.32)	0.12(0.5)
	3	0.29(0.18)	0.32(0.79)	0.61(0.84)	0.08(0.3)	0.07(0.38)	0.12(0.45)
	4	0.31(0.19)	0.37(0.87)	0.66(0.87)	0.08(0.3)	0.1(0.49)	0.13(0.44)
	5	0.3(0.18)	0.38(0.85)	0.65(0.88)	0.08(0.3)	0.1(0.47)	0.14(0.47)
	6	0.3(0.18)	0.42(0.86)	0.64(0.88)	0.08(0.3)	0.11(0.47)	0.12(0.39)
	7	0.31(0.19)	0.42(0.86)	0.6(0.89)	0.08(0.31)	0.11(0.48)	0.13(0.46)
	8	0.31(0.2)	0.46(0.86)	0.59(0.88)	0.08(0.31)	0.12(0.48)	0.12(0.4)
Left	1	0.36(0.25)	0.61(0.91)	0.34(0.32)	0.08(0.31)	0.09(0.33)	0.09(0.36)
	2	0.58(0.52)	0.35(0.33)	0.36(0.34)	0.08(0.31)	0.12(0.4)	0.1(0.38)
	3	0.45(0.39)	0.31(0.3)	0.47(0.68)	0.08(0.31)	0.11(0.4)	0.11(0.41)
	4	0.35(0.24)	0.3(0.32)	0.36(0.44)	0.08(0.31)	0.11(0.39)	0.1(0.4)
	5	0.3(0.2)	0.34(0.36)	0.28(0.3)	0.08(0.31)	0.11(0.4)	0.1(0.4)
	6	0.29(0.19)	0.43(0.83)	0.35(0.39)	0.08(0.31)	0.12(0.43)	0.09(0.38)
	7	0.28(0.18)	0.47(0.88)	0.33(0.34)	0.08(0.3)	0.11(0.39)	0.09(0.35)
	8	0.29(0.26)	0.3(0.37)	0.28(0.28)	0.07(0.3)	0.09(0.38)	0.09(0.36)
Right	1	0.41(0.26)	0.47(0.63)	0.61(0.76)	0.08(0.31)	0.08(0.31)	0.09(0.35)
	2	0.31(0.19)	0.42(0.84)	0.51(0.79)	0.08(0.31)	0.09(0.44)	0.1(0.43)
	3	0.3(0.18)	0.41(0.83)	0.61(0.82)	0.08(0.3)	0.1(0.46)	0.11(0.38)
	4	0.31(0.2)	0.45(0.83)	0.59(0.79)	0.08(0.31)	0.11(0.47)	0.11(0.36)
	5	0.31(0.19)	0.42(0.83)	0.57(0.77)	0.08(0.31)	0.11(0.48)	0.12(0.44)
	6	0.32(0.18)	0.46(0.82)	0.54(0.79)	0.08(0.3)	0.1(0.44)	0.11(0.44)
	7	0.3(0.17)	0.47(0.81)	0.53(0.8)	0.08(0.3)	0.1(0.44)	0.11(0.44)
	8	0.32(0.18)	0.47(0.8)	0.51(0.8)	0.08(0.3)	0.11(0.46)	0.12(0.45)
Baseline		**0.273(0.87)**			**0.12(0.5)**		

(a) Sequential Order of Words Discarded

$NAP_{i=98}$			$NAP_{i=4278}$		
k			k		
1	7	25	1	7	25
0.41(0.27)	0.51(0.72)	0.65(0.8)	0.08(0.31)	0.08(0.32)	0.11(0.38)
0.41(0.25)	0.55(0.8)	0.63(0.75)	0.08(0.31)	0.08(0.32)	0.12(0.39)
0.42(0.26)	0.56(0.8)	0.63(0.73)	0.08(0.31)	0.08(0.32)	0.11(0.4)
0.48(0.33)	0.56(0.81)	0.61(0.8)	0.08(0.31)	0.09(0.33)	0.13(0.47)
0.58(0.5)	0.58(0.79)	0.64(0.79)	0.08(0.31)	0.11(0.34)	0.14(0.48)
0.62(0.65)	0.56(0.75)	0.57(0.82)	0.08(0.31)	0.1(0.37)	0.11(0.41)
0.68(0.79)	0.54(0.76)	0.56(0.82)	0.09(0.32)	0.11(0.4)	0.11(0.41)
0.65(0.76)	0.56(0.8)	0.59(0.83)	0.09(0.31)	0.1(0.39)	0.11(0.41)
0.36(0.25)	0.61(0.91)	0.34(0.32)	0.08(0.31)	0.09(0.33)	0.09(0.36)
0.27(0.16)	0.6(0.68)	0.28(0.25)	0.07(0.3)	0.08(0.31)	0.08(0.3)
0.27(0.16)	0.4(0.69)	0.33(0.3)	0.07(0.3)	0.1(0.34)	0.09(0.31)
0.27(0.16)	0.48(0.68)	0.44(0.61)	0.07(0.3)	0.12(0.38)	0.1(0.34)
0.27(0.16)	0.28(0.25)	0.28(0.26)	0.07(0.3)	0.09(0.32)	0.09(0.32)
0.36(0.22)	0.31(0.29)	0.31(0.27)	0.08(0.31)	0.1(0.35)	0.1(0.36)
0.33(0.21)	0.32(0.3)	0.58(0.73)	0.08(0.31)	0.1(0.36)	0.12(0.38)
0.34(0.21)	0.33(0.32)	0.52(0.63)	0.08(0.31)	0.1(0.37)	0.11(0.38)
0.41(0.26)	0.47(0.63)	0.61(0.76)	0.08(0.31)	0.08(0.31)	0.09(0.35)
0.41(0.25)	0.53(0.8)	0.62(0.77)	0.08(0.31)	0.08(0.32)	0.11(0.33)
0.42(0.27)	0.54(0.81)	0.61(0.73)	0.08(0.31)	0.09(0.32)	0.1(0.33)
0.43(0.28)	0.51(0.81)	0.64(0.76)	0.08(0.31)	0.1(0.35)	0.11(0.38)
0.45(0.31)	0.52(0.81)	0.63(0.75)	0.08(0.31)	0.12(0.38)	0.12(0.44)
0.64(0.66)	0.49(0.56)	0.42(0.51)	0.09(0.31)	0.1(0.35)	0.08(0.35)
0.68(0.76)	0.53(0.78)	0.42(0.5)	0.09(0.31)	0.11(0.38)	0.1(0.38)
0.68(0.77)	0.52(0.71)	0.57(0.78)	0.09(0.31)	0.1(0.37)	0.11(0.39)
0.27(0.87)			**0.12(0.5)**		

(b) Sequential Order of Words Encoded

Table 5.9: The performances observed over the $\{\mathrm{T}\}^{\mathrm{Y_A T_E A}}_{\mathrm{Y_A T_E A}}$ when $|R_s| = 100$ and similarities are computed using the *Euclidean* distance.

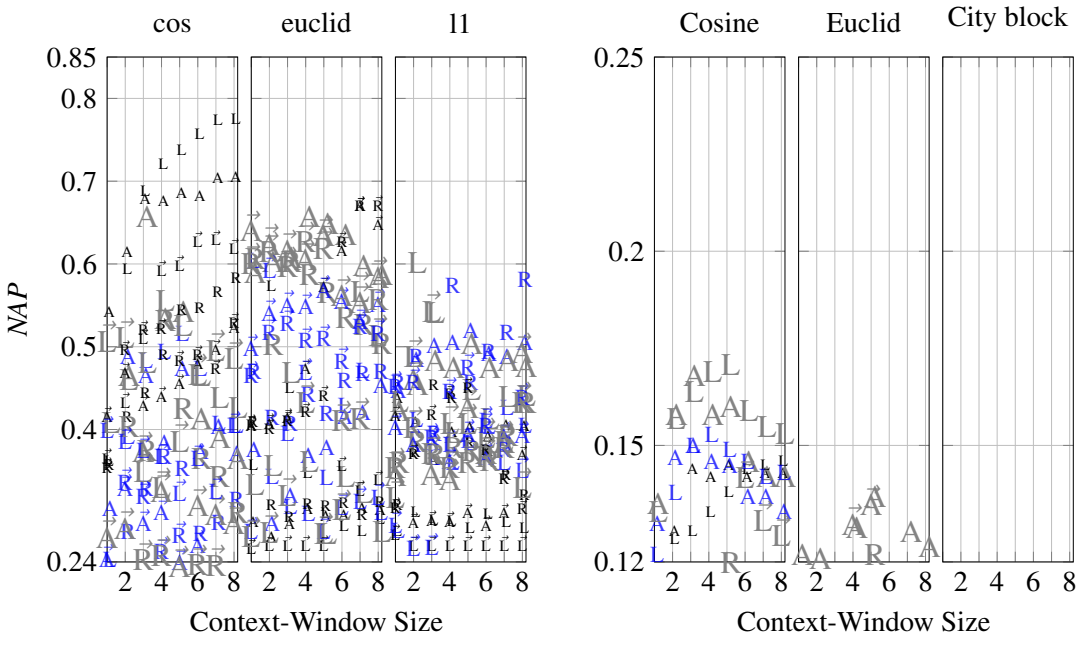
Context		$NAP_{i=98}$			$NAP_{i=4278}$		
dir	size	k			k		
		1	7	25	1	7	25
Around	1	0.44(0.33)	0.45(0.59)	0.43(0.62)	0.08(0.31)	0.07(0.34)	0.07(0.35)
	2	0.41(0.43)	0.49(0.7)	0.46(0.65)	0.08(0.31)	0.09(0.33)	0.09(0.33)
	3	0.37(0.24)	0.51(0.73)	0.49(0.73)	0.08(0.31)	0.09(0.32)	0.09(0.33)
	4	0.4(0.24)	0.51(0.77)	0.48(0.76)	0.08(0.31)	0.09(0.34)	0.1(0.34)
	5	0.41(0.24)	0.52(0.76)	0.51(0.79)	0.08(0.31)	0.1(0.35)	0.11(0.36)
	6	0.4(0.27)	0.49(0.77)	0.48(0.79)	0.08(0.31)	0.1(0.35)	0.1(0.36)
	7	0.41(0.27)	0.49(0.77)	0.49(0.79)	0.08(0.31)	0.1(0.36)	0.1(0.36)
	8	0.41(0.28)	0.51(0.76)	0.48(0.8)	0.08(0.31)	0.1(0.36)	0.11(0.37)
Left	1	0.28(0.23)	0.28(0.23)	0.44(0.55)	0.06(0.26)	0.06(0.26)	0.09(0.31)
	2	0.3(0.17)	0.42(0.61)	0.61(0.82)	0.08(0.3)	0.08(0.32)	0.09(0.33)
	3	0.29(0.16)	0.37(0.5)	0.54(0.78)	0.08(0.3)	0.08(0.31)	0.08(0.32)
	4	0.29(0.17)	0.36(0.54)	0.4(0.57)	0.08(0.3)	0.07(0.31)	0.09(0.35)
	5	0.28(0.16)	0.39(0.57)	0.45(0.67)	0.08(0.3)	0.08(0.31)	0.09(0.32)
	6	0.3(0.17)	0.41(0.59)	0.42(0.66)	0.08(0.3)	0.08(0.31)	0.08(0.32)
	7	0.29(0.17)	0.43(0.6)	0.44(0.71)	0.08(0.3)	0.08(0.31)	0.1(0.37)
	8	0.28(0.16)	0.41(0.62)	0.44(0.72)	0.08(0.3)	0.08(0.32)	0.1(0.37)
Right	1	0.31(0.19)	0.46(0.41)	0.36(0.6)	0.08(0.3)	0.08(0.31)	0.07(0.32)
	2	0.38(0.25)	0.49(0.69)	0.39(0.56)	0.08(0.31)	0.09(0.34)	0.08(0.33)
	3	0.46(0.41)	0.4(0.51)	0.37(0.5)	0.08(0.31)	0.08(0.32)	0.07(0.32)
	4	0.44(0.31)	0.58(0.65)	0.38(0.49)	0.08(0.31)	0.08(0.32)	0.08(0.31)
	5	0.39(0.23)	0.46(0.57)	0.38(0.51)	0.08(0.31)	0.09(0.34)	0.08(0.31)
	6	0.38(0.22)	0.5(0.63)	0.37(0.49)	0.08(0.31)	0.09(0.32)	0.08(0.31)
	7	0.35(0.19)	0.52(0.6)	0.4(0.53)	0.08(0.31)	0.08(0.31)	0.08(0.31)
	8	0.32(0.18)	0.58(0.74)	0.44(0.62)	0.08(0.3)	0.08(0.31)	0.09(0.32)
Baseline		**0.273(0.87)**			**0.12(0.5)**		

(a) Sequential Order of Words Discarded

$NAP_{i=98}$			$NAP_{i=4278}$		
k			k		
1	7	25	1	7	25
0.42(0.27)	0.41(0.61)	0.35(0.59)	0.08(0.31)	0.07(0.34)	0.08(0.36)
0.29(0.16)	0.39(0.53)	0.47(0.62)	0.08(0.3)	0.08(0.32)	0.08(0.33)
0.3(0.16)	0.39(0.48)	0.35(0.43)	0.08(0.3)	0.08(0.3)	0.07(0.3)
0.29(0.17)	0.37(0.49)	0.35(0.44)	0.08(0.3)	0.08(0.3)	0.07(0.3)
0.3(0.17)	0.4(0.51)	0.44(0.54)	0.08(0.3)	0.08(0.31)	0.08(0.3)
0.29(0.19)	0.37(0.43)	0.38(0.41)	0.08(0.31)	0.07(0.3)	0.07(0.3)
0.3(0.18)	0.4(0.45)	0.39(0.45)	0.08(0.3)	0.08(0.3)	0.07(0.3)
0.37(0.41)	0.4(0.47)	0.5(0.56)	0.08(0.31)	0.08(0.31)	0.08(0.31)
0.28(0.23)	0.28(0.23)	0.44(0.55)	0.06(0.26)	0.06(0.26)	0.09(0.31)
0.27(0.16)	0.26(0.17)	0.5(0.68)	0.07(0.3)	0.07(0.3)	0.1(0.33)
0.27(0.16)	0.26(0.16)	0.55(0.69)	0.07(0.3)	0.07(0.3)	0.08(0.31)
0.27(0.16)	0.39(0.58)	0.42(0.56)	0.07(0.3)	0.1(0.37)	0.08(0.3)
0.27(0.16)	0.39(0.51)	0.42(0.56)	0.07(0.3)	0.08(0.31)	0.08(0.31)
0.27(0.16)	0.42(0.49)	0.4(0.54)	0.07(0.3)	0.08(0.3)	0.08(0.3)
0.27(0.16)	0.37(0.49)	0.4(0.55)	0.07(0.3)	0.08(0.3)	0.08(0.31)
0.27(0.16)	0.36(0.51)	0.34(0.44)	0.07(0.3)	0.08(0.31)	0.08(0.33)
0.31(0.19)	0.46(0.41)	0.36(0.6)	0.08(0.3)	0.08(0.31)	0.07(0.32)
0.38(0.24)	0.46(0.68)	0.41(0.68)	0.08(0.31)	0.1(0.37)	0.08(0.33)
0.42(0.38)	0.4(0.52)	0.38(0.49)	0.08(0.31)	0.08(0.32)	0.07(0.32)
0.45(0.32)	0.45(0.59)	0.36(0.46)	0.08(0.31)	0.09(0.33)	0.07(0.31)
0.46(0.47)	0.48(0.49)	0.37(0.45)	0.08(0.31)	0.08(0.31)	0.07(0.31)
0.38(0.24)	0.41(0.5)	0.38(0.51)	0.08(0.31)	0.08(0.31)	0.07(0.31)
0.35(0.2)	0.38(0.48)	0.39(0.5)	0.08(0.31)	0.08(0.31)	0.07(0.31)
0.31(0.18)	0.45(0.57)	0.44(0.59)	0.08(0.3)	0.08(0.32)	0.08(0.32)
0.27(0.87)			**0.12(0.5)**		

(b) Sequential Order of Words Encoded

Table 5.10: The performances observed over the $\{\mathrm{T}\}^{\mathrm{Y_A T_E A}}_{\mathrm{Y_A T_E A}}$ when $|R_s| = 100$ and similarities are computed using the *city block* distance.



(a) $NAP_{i=98}$ (b) $NAP_{i=4278}$

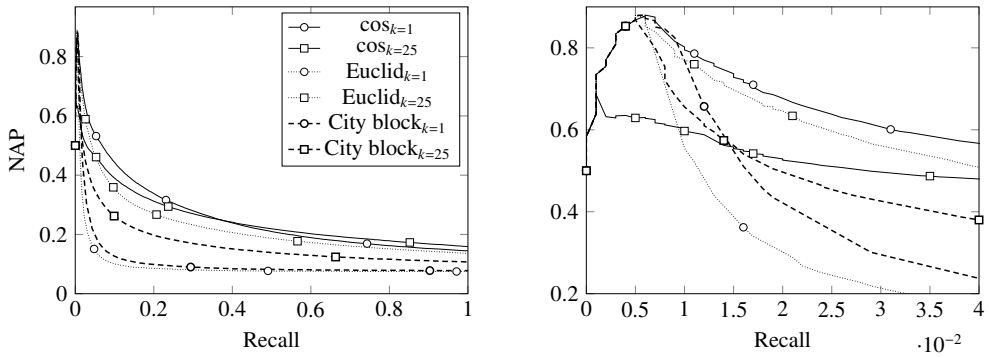
Figure 5.11: The performances observed in $\{\mathrm{T}\}^{\mathrm{YATEA}}_{\mathrm{YATEA}}$. The performance is reported using NAP_i = 98 (i.e., for 2% recall) and NAP_i = 4278 (i.e., 100% recall). Similar representation format as Figure 5.8 is used.

In addition to *I kappa B*, $\{\mathrm{T}\}^{\mathrm{YATEA}}_{\mathrm{YATEA}}$ contains candidate terms such as *control of I kappa B-alpha proteolysis*, *inhibitor I kappa B*, *endogenous I kappa B*, and so on, of which many are invalid terms.[1] Since these candidate terms share a similar context with valid protein terms, they also appear at the top of the obtained ranked sets. As a result, their presence in the list of candidate terms deteriorate the method's performance. As suggested previously, combining a unithood measure with the score generated by the classification method can help to alleviate these errors, particularly for large recall values.

Independently of the direction in which context-windows are extended, for the nearest neighbour (i.e., $k = 1$), a positive correlation between the size of context-window and the performance of the method (both for detecting protein terms and valid candidate terms) is observable. However, when $k = 25$ and the context-windows are larger than three tokens, then a low negative correlation between the size of context-windows and the performance of the method is observable. In these experiments, when the Euclidean or the city block distance are employed, then using $k = 25$ results in a more stable performance across different cut-off points for computing NAP_i than $k = 1$. As shown in Figure 5.12, if the goal is to extract only a small fraction of protein terms (e.g., 100), then using the city block or the Euclidean distance in the nearest neighbour framework gives the best performance. However, the performance of these combinations drops abruptly for larger recall values.

Similar to the previous experiment, encoding information about the sequential order

[1] For instance, in $\{\mathrm{T}\}^{\mathrm{YATEA}}_{\mathrm{YATEA}}$, *I kappa B* appears nested in 221 terms.



(a) (b)

Figure 5.12: The performances observed in $\{\mathrm{T}\}^{\mathrm{Y_A T_E A}}_{\mathrm{Y_A T_E A}}$ Using $|R_s| = 100$: NAP_i for $1 \leq i \leq 4278$ (i.e, various recall points) computed for context-windows of size 5 tokens that extend around candidate terms. Shown are results obtained for $k = 1, 25$ and the three similarity scores—that is, the cosine measure, the Euclidean and the city block distance; (a) shows NAP for $1 \leq i \leq 4278$, whereas (b) details the results for $1 \leq i \leq 172$ (i.e, recall less than 4%).

of words in the context-windows does not improve the performances, particularly when using the cosine measure or the city block distance. Moreover, likewise experiments in $\{\mathrm{T}\}^{\text{c-}value}_{\text{ideal}}$, models constructed by collecting co-occurrences from context-windows that extend to left side of candidate terms, on average, show the best performances. However, using context-windows that extend around terms is a more cautious choice than choosing context-windows extending to the left or right side of candidate terms in the sense that they show less variance when parameters of the method, including the employed similarity metric, change.

Similar to the experiments over $\{\mathrm{T}\}^{\text{c-}value}_{\text{ideal}}$, the choice for the context-window's size remains dependant on the choices that are made for selecting the rest of the method's parameters. If the method's performance is averaged over k, then using context-windows of size 3 to 6 tokens is recommend. If $k = 1$, larger context-windows result in better performance; however, if k is large, then extending context-windows to more than 6 tokens reduces the performance. To investigate the similarity of the impact of the context-window's size on the method's performance between experiments in $\{\mathrm{T}\}^{\mathrm{Y_A T_E A}}_{\mathrm{Y_A T_E A}}$ and $\{\mathrm{T}\}^{\text{c-}value}_{\text{ideal}}$, Table 5.11 reports the Spearman's coefficient correlation (r_s) when the size of context-windows is considered as the ranking variable.

In these tables, the results obtained in $\{\mathrm{T}\}^{\mathrm{Y_A T_E A}}_{\mathrm{Y_A T_E A}}$ and $\{\mathrm{T}\}^{\text{c-}value}_{\text{ideal}}$ are compared when the method's parameters, except the size of context-windows, are fixed. Each cell of Tables 5.11a and 5.11b shows the computed r_s between each column in Tables 5.8, 5.9, and 5.10 (from experiments in $\{\mathrm{T}\}^{\mathrm{Y_A T_E A}}_{\mathrm{Y_A T_E A}}$) and the corresponding column in Tables 5.5, 5.6, and 5.7. Accordingly, if the choice for the best performing size of context-windows is similar in $\{\mathrm{T}\}^{\mathrm{Y_A T_E A}}_{\mathrm{Y_A T_E A}}$ and $\{\mathrm{T}\}^{\text{c-}value}_{\text{ideal}}$, then a high correlation (i.e., 1) is expected. As shown in Table 5.11, a high-correlation is observed only when using the cosine measure—that is, the same size of context-windows in both experiments results in a high performance.

Simialrity Metric	k	A	L	R	$\vec{A}$	$\vec{L}$	$\vec{R}$
Cosine	**1**	0.36	0.81	0.55	0.9	0.89	0.62
	7	0.83	0.4	0.81	0.48	0.05	0.07
	25	0.81	0.12	0.36	0.67	-0.36	0.33
Euclid	**1**	-0.4	0.9	-0.25	-0.6	-0.14	-0.07
	7	-0.88	0.19	0.41	0.36	0.62	0.0
	25	0.57	-0.32	-0.31	-0.4	0.6	-0.5
City block	**1**	0.33	0.01	0.4	-0.07	0.58	0.57
	7	0.17	0.6	0.46	0.11	-0.33	-0.55
	25	-0.5	0.4	0.43	0.05	0.78	-0.6

(a) Using *NAP* at 2% recall.

Simialrity Metric	k	A	L	R	$\vec{A}$	$\vec{L}$	$\vec{R}$
Cosine	**1**	0.6	0.83	0.97	0.21	0.28	0.14
	7	0.61	0.21	0.73	0.67	-0.17	0.33
	25	0.71	0.01	0.96	0.58	-0.71	0.69
Euclid	**1**	-0.85	0.29	-0.41	-0.24	-0.52	-0.34
	7	-0.86	0.53	0.19	-0.38	0.19	-0.76
	25	-0.29	0.44	0.76	0.6	0.24	-0.11
City block	**1**	0.18	-0.19	-0.52	-0.22	-0.58	-0.25
	7	0.65	-0.18	-0.08	0.5	-0.1	-0.16
	25	-0.43	-0.09	-0.14	0.08	0.75	-0.23

(b) Using *NAP* at 100% recall.

Table 5.11: Spearman's correlation coefficient (r_s) between the results obtained in $\{\mathrm{T}\}^{c\text{-}value}_{\text{ideal}}$ and $\{\mathrm{T}\}^{\text{YATEA}}_{\text{YATEA}}$ when the context-window's size used as the ranking variable and the remaining method's parameters are fixed. Tables (a) and (b) show the observed r_s when performances are computed using *NAP* at 2% and 100% recall, respectively.

5.4.3 Corpus Size: The Bigger the Better?

As described earlier, independent of the context-window's configuration for collecting co-occurrences, due to the Zipfian distribution of terms and words in context-windows, vectors that represent candidate terms are inevitably high-dimensional and *sparse* (i.e., most of the elements of vectors are zero). Whereas the high-dimensionality of vectors hinders the computation of similarities, their sparseness is likely to diminish the discriminatory power of the constructed distributional model. To overcome the high-dimensionality barrier, random projections are employed in this research in order to reduce the dimension of vectors to a fixed certain size. Now that the vectors' dimension is set to a constant size, it is hypothesised that enlarging the size of the corpus reduces the number of zero elements in the vectors, and thus, the performance of the distributional model improves (e.g., see Bullinaria and Levy (2007), Pantel et al. (2009) as well as Gorman and Curran (2006)).

In this section, the interplay between the size of the corpus and choosing the most discriminating configuration for context-windows in the proposed classification task is investigated. Two questions are investigated using empirical experiments, including (a) whether increasing the size of the corpus that is used for collecting co-occurrence frequencies enhances the performance of the classification task and (b) how doing so influences the choices that are made for configuring context-windows. The GENIA corpus is thus enlarged by fetching 223,316 abstracts from the PubMed repository,[1] of which each

[1] Accessible at http://www.ncbi.nlm.nih.gov/pmc/tools/ftp/#Source_files.

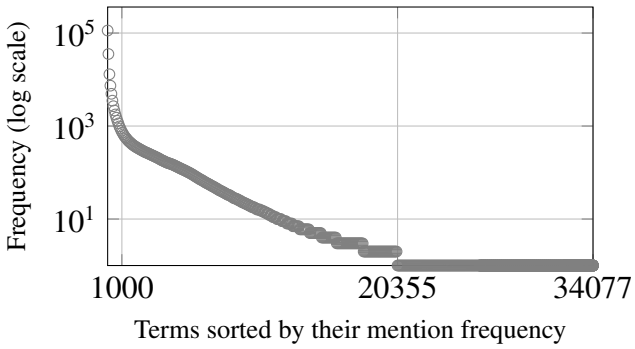


Figure 5.13: The frequency of terms in the enlarged corpus: more than 40% of the terms never appear in the enlarged corpus.

abstract contains at least three of the terms in the employed terminological resource.[1] Similar to the previous experiments, besides normalising text to lower-case letters and a Penn Treebank tokenisation, no other text preprocessing is performed. As a result, the enlarged corpus has more than 55 million tokens and a vocabulary of size 881,040.

This enlarged corpus contains 9,179,046 additional mentions of the terms ranked in the $\{T\}^{\text{c-}value}_{\text{ideal}}$. As expected, the term frequencies in the enlarged corpus has a long tail distribution—that is, a small number of terms are frequent whereas the majority of terms are mentioned a few times (Figure 5.13). Using the vector space construction method explained in Section 5.2.1, the constructed vectors from $\{T\}^{\text{c-}value}_{\text{ideal}}$ are augmented by the collected co-occurrences from the enlarged corpus. Hereafter, this set of vectors is denoted by $\{T\}^{\text{c-}value}_{\text{Enlarged}}$. Nevertheless, the obtained vectors in $\{T\}^{\text{c-}value}_{\text{Enlarged}}$ are less sparse than the previously built vectors in $\{T\}^{\text{c-}value}_{\text{ideal}}$. For example, in $\{T\}^{\text{c-}value}_{\text{ideal}}$, vectors that are constructed by collecting the co-occurrence frequencies from context-windows that extend by the size of one token around terms are approximately *five times* sparser than vectors in $\{T\}^{\text{c-}value}_{\text{Enlarged}}$ that are constructed by collecting the co-occurrence frequencies from context-windows of the same configuration.[2]

An identical process employed for term classification in $\{T\}^{\text{c-}value}_{\text{ideal}}$ (see, Section 5.4.1) is employed in $\{T\}^{\text{c-}value}_{\text{Enlarged}}$. 48 different vector spaces are constructed, each reflects one of the possible combinations for context-window's configuration. The classification is then performed using three values of k (i.e., $k = 1, 7, 25$) and the same set of reference vectors (R_s) employed in the experiments reported in Section 5.4.1 (i.e., R_s comprises 100 terms of which 36 are positive examples). Tables 5.12, 5.13, and 5.14 report the observed results when the cosine measure, Euclidean distance, and the city block distance are employed for computing similarities, respectively. Figure 5.14 plots numbers reported in these tables.

[1] The set of employed abstracts can be retrieved from http://atmykitchen.info/phd/materials/genia/extended_abstracts.tar.gz.

[2] Please note that in the proposed method, apart from the size of the corpus employed for collecting

Context		$NAP_{i=200}$			$NAP_{i=8900}$		
dir	size	k			k		
		1	7	25	1	7	25
Around	1	0.817	0.724	0.714	0.371	0.388	0.389
	2	0.61	0.797	0.748	0.395	0.409	0.41
	3	0.641	0.69	0.646	0.41	0.416	0.414
	4	0.672	0.795	0.656	0.417	0.427	0.421
	5	0.686	0.794	0.659	0.416	0.428	0.422
	6	0.673	0.781	0.686	0.414	0.428	0.421
	7	0.688	0.797	0.723	0.416	0.43	0.42
	8	0.698	0.805	0.692	0.416	0.427	0.415
Left	1	0.643	0.716	0.772	0.333	0.372	0.356
	2	0.789	0.81	0.782	0.377	0.401	0.388
	3	0.834	0.798	0.808	0.398	0.421	0.417
	4	0.851	0.825	0.772	0.415	0.435	0.437
	5	0.838	0.823	0.781	0.42	0.445	0.451
	6	0.825	0.82	0.807	0.427	0.453	0.458
	7	0.813	0.823	0.833	0.426	0.456	0.461
	8	0.808	0.85	0.82	0.428	0.456	0.461
Right	1	0.765	0.655	0.609	0.349	0.383	0.361
	2	0.611	0.819	0.748	0.359	0.389	0.38
	3	0.664	0.763	0.641	0.374	0.39	0.382
	4	0.681	0.812	0.648	0.385	0.395	0.388
	5	0.692	0.837	0.707	0.389	0.397	0.389
	6	0.684	0.828	0.705	0.389	0.395	0.39
	7	0.708	0.834	0.727	0.39	0.392	0.389
	8	0.712	0.844	0.726	0.39	0.392	0.387
Baseline		**0.364**			**0.273**		

(a) Sequential Order of Words Discarded

$NAP_{i=200}$			$NAP_{i=8900}$		
k			k		
1	7	25	1	7	25
0.748	0.687	0.608	0.357	0.388	0.365
0.703	0.7	0.614	0.369	0.394	0.374
0.691	0.733	0.555	0.373	0.398	0.378
0.676	0.788	0.591	0.375	0.399	0.381
0.657	0.793	0.575	0.376	0.395	0.379
0.642	0.778	0.561	0.373	0.395	0.377
0.63	0.805	0.58	0.372	0.394	0.375
0.625	0.831	0.592	0.372	0.395	0.373
0.643	0.716	0.772	0.333	0.372	0.356
0.574	0.868	0.898	0.323	0.414	0.424
0.711	0.949	0.906	0.347	0.432	0.423
0.731	0.921	0.897	0.373	0.428	0.42
0.777	0.905	0.912	0.369	0.417	0.412
0.806	0.919	0.833	0.384	0.407	0.399
0.843	0.889	0.843	0.374	0.393	0.388
0.845	0.854	0.767	0.371	0.371	0.382
0.765	0.655	0.609	0.349	0.383	0.361
0.747	0.657	0.647	0.361	0.385	0.368
0.744	0.674	0.598	0.369	0.386	0.37
0.727	0.707	0.589	0.369	0.384	0.37
0.698	0.732	0.583	0.368	0.383	0.368
0.68	0.761	0.62	0.366	0.386	0.368
0.666	0.762	0.643	0.365	0.383	0.365
0.651	0.749	0.617	0.365	0.38	0.362
0.364			**0.273**		

(b) Sequential Order of Words Encoded

Table 5.12: The performances observed over the $\{\mathrm{T}\}^{\text{c-}value}_{\text{Enlarged}}$ when $|R_s| = 100$ and similarities are computed using the *cosine* between vectors. Similar to the experiments over $\{\mathrm{T}\}^{\text{c-}value}_{\text{ideal}}$, the performance is shown with regards to the observed NAP_i, for $i = 200$ (i.e., recall = 0.02) and $i = 8900$ (i.e., recall = 1.0).

Context		$NAP_{i=200}$			$NAP_{i=8900}$		
dir	size	k			k		
		1	7	25	1	5	25
Around	1	0.595	0.772	0.742	0.341	0.312	0.329
	2	0.639	0.771	0.852	0.349	0.331	0.315
	3	0.692	0.752	0.728	0.359	0.325	0.302
	4	0.627	0.773	0.71	0.302	0.322	0.298
	5	0.621	0.732	0.814	0.333	0.313	0.319
	6	0.623	0.783	0.767	0.345	0.32	0.328
	7	0.598	0.728	0.742	0.33	0.316	0.31
	8	0.59	0.736	0.76	0.322	0.316	0.306
Left	1	0.605	0.82	0.73	0.314	0.343	0.314
	2	0.593	0.777	0.706	0.322	0.338	0.319
	3	0.635	0.793	0.637	0.334	0.344	0.323
	4	0.66	0.81	0.632	0.339	0.34	0.315
	5	0.681	0.788	0.662	0.343	0.341	0.311
	6	0.655	0.831	0.667	0.338	0.334	0.31
	7	0.64	0.772	0.633	0.336	0.328	0.304
	8	0.629	0.772	0.648	0.335	0.322	0.303
Right	1	0.638	0.854	0.66	0.287	0.299	0.301
	2	0.552	0.85	0.599	0.282	0.305	0.282
	3	0.641	0.773	0.697	0.296	0.3	0.293
	4	0.556	0.713	0.57	0.279	0.299	0.283
	5	0.522	0.744	0.565	0.275	0.301	0.283
	6	0.552	0.729	0.773	0.28	0.309	0.298
	7	0.542	0.708	0.74	0.276	0.309	0.312
	8	0.544	0.727	0.633	0.276	0.313	0.293
Baseline		**0.364**			**0.273**		

(a) Sequential Order of Words Discarded

$NAP_{i=200}$			$NAP_{i=8900}$		
k			k		
1	7	25	1	5	25
0.682	0.886	0.668	0.292	0.304	0.302
0.694	0.886	0.686	0.296	0.313	0.311
0.698	0.897	0.696	0.297	0.317	0.319
0.683	0.893	0.698	0.296	0.322	0.323
0.672	0.898	0.71	0.291	0.325	0.322
0.657	0.906	0.709	0.292	0.331	0.322
0.618	0.894	0.71	0.286	0.332	0.322
0.609	0.867	0.709	0.282	0.33	0.317
0.605	0.82	0.73	0.314	0.343	0.314
0.646	0.749	0.77	0.335	0.301	0.323
0.663	0.844	0.748	0.327	0.324	0.333
0.694	0.837	0.748	0.344	0.331	0.331
0.743	0.833	0.757	0.351	0.329	0.336
0.716	0.801	0.831	0.347	0.328	0.332
0.742	0.776	0.833	0.349	0.322	0.332
0.693	0.842	0.832	0.337	0.328	0.323
0.638	0.854	0.66	0.287	0.299	0.301
0.634	0.88	0.68	0.285	0.302	0.304
0.623	0.879	0.679	0.286	0.309	0.309
0.635	0.862	0.686	0.285	0.31	0.316
0.638	0.847	0.698	0.282	0.311	0.315
0.635	0.855	0.702	0.284	0.318	0.317
0.603	0.877	0.707	0.281	0.32	0.318
0.582	0.84	0.696	0.279	0.323	0.318
0.364			**0.273**		

(b) Sequential Order of Words Encoded

Table 5.13: The results observed in $\{\mathrm{T}\}^{\text{c-}value}_{\text{Enlarged}}$ when $|R_s| = 100$ and similarities are computed using the *Euclidean* distance.

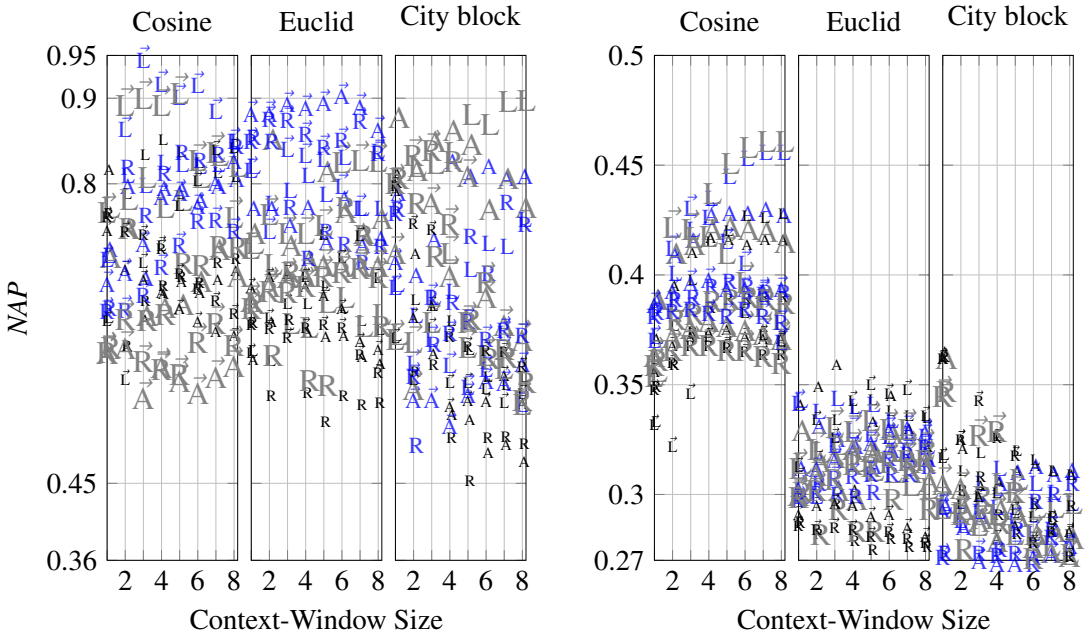
Context dir	Context size	$NAP_{i=200}$ k=1	$NAP_{i=200}$ k=7	$NAP_{i=200}$ k=25	$NAP_{i=8900}$ k=1	$NAP_{i=8900}$ k=5	$NAP_{i=8900}$ k=25
Around	1	0.812	0.789	0.878	0.365	0.295	0.362
	2	0.718	0.693	0.816	0.321	0.285	0.293
	3	0.601	0.735	0.85	0.304	0.29	0.289
	4	0.53	0.826	0.862	0.306	0.305	0.293
	5	0.524	0.809	0.846	0.311	0.31	0.29
	6	0.486	0.821	0.77	0.313	0.313	0.282
	7	0.495	0.807	0.806	0.311	0.309	0.283
	8	0.475	0.809	0.772	0.309	0.311	0.28
Left	1	0.623	0.683	0.624	0.318	0.296	0.296
	2	0.665	0.646	0.82	0.311	0.286	0.293
	3	0.658	0.667	0.824	0.319	0.305	0.296
	4	0.64	0.675	0.837	0.327	0.312	0.305
	5	0.606	0.7	0.881	0.32	0.305	0.307
	6	0.575	0.731	0.87	0.316	0.305	0.294
	7	0.565	0.727	0.896	0.311	0.301	0.294
	8	0.575	0.758	0.897	0.309	0.296	0.296
Right	1	0.801	0.77	0.806	0.363	0.273	0.347
	2	0.581	0.495	0.777	0.302	0.254	0.275
	3	0.589	0.613	0.715	0.308	0.267	0.299
	4	0.503	0.661	0.826	0.291	0.275	0.287
	5	0.453	0.739	0.642	0.282	0.284	0.267
	6	0.503	0.696	0.596	0.296	0.297	0.267
	7	0.505	0.778	0.597	0.29	0.3	0.266
	8	0.496	0.753	0.568	0.295	0.305	0.265
Baseline		0.364			0.273		

(a) Sequential Order of Words Discarded

$NAP_{i=200}$ k=1	$NAP_{i=200}$ k=7	$NAP_{i=200}$ k=25	$NAP_{i=8900}$ k=1	$NAP_{i=8900}$ k=5	$NAP_{i=8900}$ k=25
0.795	0.712	0.725	0.363	0.294	0.307
0.647	0.551	0.565	0.326	0.266	0.305
0.661	0.551	0.783	0.3	0.271	0.29
0.542	0.521	0.698	0.281	0.27	0.281
0.55	0.557	0.815	0.287	0.27	0.297
0.548	0.574	0.673	0.28	0.273	0.278
0.579	0.571	0.64	0.284	0.274	0.276
0.546	0.593	0.623	0.284	0.278	0.274
0.623	0.683	0.624	0.318	0.296	0.296
0.574	0.594	0.62	0.297	0.29	0.294
0.635	0.623	0.644	0.3	0.294	0.288
0.57	0.592	0.725	0.302	0.296	0.293
0.571	0.566	0.615	0.294	0.29	0.284
0.564	0.588	0.605	0.291	0.284	0.284
0.547	0.629	0.59	0.29	0.286	0.279
0.558	0.546	0.548	0.283	0.277	0.275
0.801	0.77	0.806	0.363	0.273	0.347
0.756	0.579	0.839	0.326	0.256	0.293
0.749	0.672	0.845	0.344	0.275	0.329
0.624	0.629	0.758	0.327	0.274	0.331
0.613	0.643	0.644	0.319	0.275	0.303
0.608	0.623	0.691	0.279	0.27	0.273
0.607	0.633	0.612	0.284	0.282	0.267
0.589	0.627	0.553	0.273	0.266	0.265
0.364			0.273		

(b) Sequential Order of Words Encoded

Table 5.14: The results observed in $\{T\}^{c\text{-}value}_{Enlarged}$ when $|R_s| = 100$ and similarities are computed using the *city block* distance.



(a) $NAP_{i=200}$ (b) $NAP_{i=8900}$

Figure 5.14: The NAP_i observed over $\{T\}^{c\text{-}value}_{Enlarged}$ for $i = 200$ (i.e., 2% recall) and $i = 8900$ (i.e., recall 100%) shown in (a) and (b), respectively.

As shown in Figure 5.14b, likewise the previous experiments and independently of the size of the input corpus, if *NAP* is computed for a high recall such as 100%, then the cosine similarity outperforms both the Euclidean and City block distance. In $\{\mathrm{T}\}^{\text{c-}value}_{\text{ideal}}$ and for $NAP_{i=8900}$, the best classification performance is observed using models that are built by collecting co-occurrence frequencies in context-windows of size 4 or 5 tokens that extend around terms. However, in $\{\mathrm{T}\}^{\text{c-}value}_{\text{Enlarged}}$, models that are built using context-windows that expand to the left side of the candidate terms outperform models that are built by collecting co-occurrence frequencies in context-windows that expand around the terms. In addition, similar to the experiments in $\{\mathrm{T}\}^{\text{c-}value}_{\text{ideal}}$, in $\{\mathrm{T}\}^{\text{c-}value}_{\text{Enlarged}}$, a large value for k results in a more desirable performance than a small value such as $k = 1$, too.

Figure 5.15 plots the changes that are observed by enlarging the size of the input corpus when the performance is measured using $NAP_{i=8900}$. For instance, when using the cosine similarity and $k = 25$ in the model constructed using context-windows of size 6 tokens that neglect word order information and extend only to the *left* side of terms, the $NAP_{i=8900}$ in $\{\mathrm{T}\}^{\text{c-}value}_{\text{Enlarged}}$ is 0.461 (see Table 5.12). However, the same classification parameters and configuration for context-windows in $\{\mathrm{T}\}^{\text{c-}value}_{\text{ideal}}$ gives the performance of 0.405 (see Table 5.5). In Figure 5.15, this increase in the performance is marked by a wide circle at the corresponding position. Accordingly, the plotted results suggest that when the corpus size increases, the type of employed similarity measure plays a role in determining the changes in the performances. When similarities are calculated using the cosine measure, enlarging the size of the corpus enhances the performance. Similarly, the city block distance shows a relatively better performance with larger input corpus. However, when similarities are measured using the Euclidean distance, an increase in the size of the corpus can drastically decline the performance.

Figures 5.14a and 5.16, similar to Figures 5.14b and 5.15, show the method's performance, however, when it is measured by $NAP_{i=200}$ (i.e., for the 2% recall). As shown, if the performance is assessed for a small recall, then all three measures equally perform well. In this case, increasing the size of the corpus enhances or diminishes the performance by approximately 20%. Again, the Euclidean distance is more susceptible to an increase in the corpus size. In contrary, the cosine measure consistently shows a better performance when the corpus size increases. Results suggest a similar conclusion for the city block distance. Although in this case, the enhancement is not as steady as the cosine measure and it depends on the value of k and the context-window's configuration, too. Particularly, for $k = 1$, the performance frequently drops when the size of the input corpus increases.

In the experiments over $\{\mathrm{T}\}^{\text{c-}value}_{\text{Enlarged}}$, with respect to the relationship between recall and performance, the behaviour of similarity measures is similar to the previous experiments. If the performance of the method is studied across recall values, city block distance outperforms the cosine measure then for a small recall. When using the city block distance, however, as shown in Figure 5.17, the performance drops abruptly as recall increases. Compared to Figure 5.10 and 5.12, for a number of context-window configurations, enlarging the corpus eminently enhances the performance of the cosine metric at small recall

---

co-occurrences, the sparseness of vectors is also determined by the number of zero and non-zero elements in word vectors.

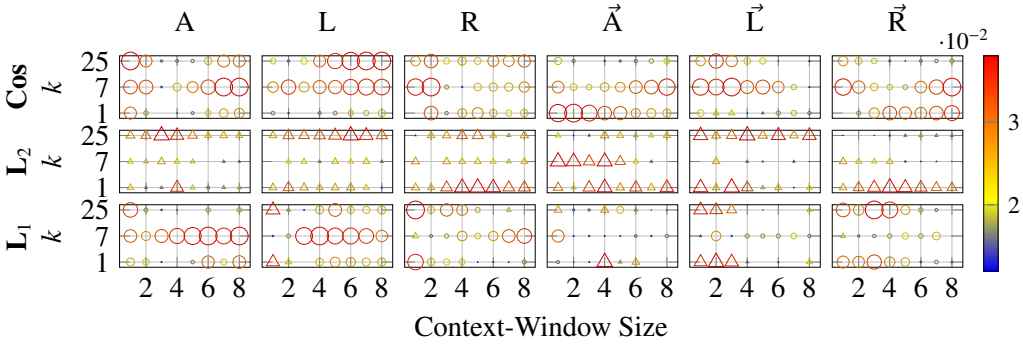


Figure 5.15: Changes in the performance of models caused by increasing the size of the input corpus. The method's performance is measured using *NAP* at 100% recall. The figure shows the absolute value of the difference between the performance obtained from a model constructed in $\{T\}^{\text{c-value}}_{\text{ideal}}$ and the corresponding model in $\{T\}^{\text{c-value}}_{\text{Enlarged}}$. Triangles denote negative change, while circles show positive change. The size/colour of shapes represents the amount of changes. The x-axis shows various configurations of context-windows (i.e., size, direction, and encoding word order information). The y-axis, however, represents classification parameters (i.e., the values of k and the employed measures for calculating similarities).

values and thus makes it a rival to the Euclidean and city block distance in tasks that aim for extracting a small number of terms.

Similar to Table 5.11, Table 5.15 reports the Spearman's coefficient correlation (r_s) when the size of context-window is considered as the ranking variable (see page 166) and the results obtained in $\{T\}^{\text{c-value}}_{\text{Enlarged}}$ are compared with the results in $\{T\}^{\text{c-value}}_{\text{ideal}}$. A high positive correlation for the choice of the best performing sizes of context-windows is observable between these two experiments only when cosine is employed for computing similarities and the performance is assessed for a large recall value. Otherwise, as the Table 5.15 suggests, if the size of the corpus changes, unfortunately the choice for the

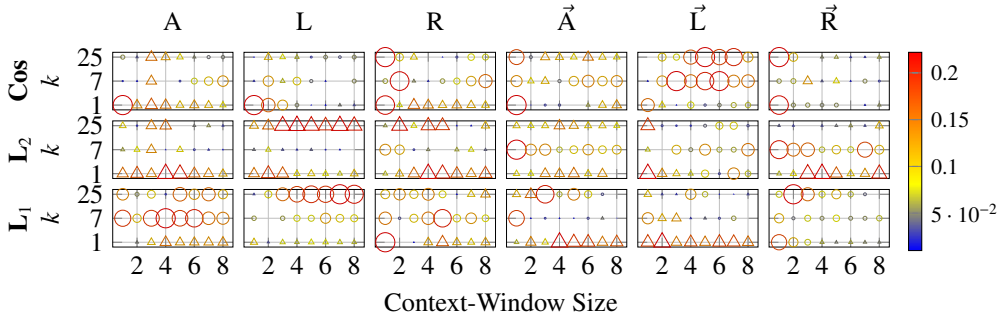


Figure 5.16: The changes in the performance of the method caused by increasing the size of the input corpus when the method's performance is measured using $NAP_{i=200}$ (i.e, 2% recall). The presentation format is similar to Figure 5.15: circles show positive effect whereas triangles show negative impact on the performance.

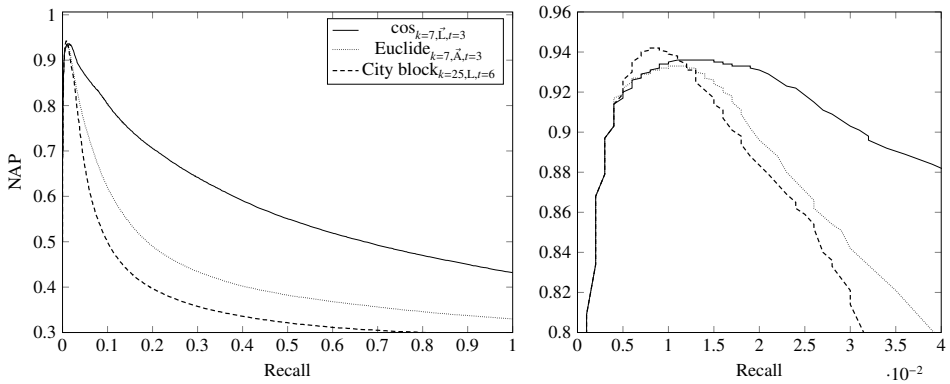


Figure 5.17: Performance of the similarity metrics over the range of recall values: shown the observed performances of the top performing models in the reported results. Similar to the previous experiments, when using the Euclidean or city block distance, the performance drops abruptly. However, if the aim is to extract only a small number of terms such as 100 (i.e., in this example, approximately a recall less than 1%), then the city block distance outperforms other similarity metrics.

size of context-window must be revised to achieve the best performance.

With respect to the effect of encoding information about the sequential order of words in the models, the observed results in $\{T\}^{c\text{-}value}_{Enlarged}$ (Figure 5.18) is also inconsistent with the observations that are made over $\{T\}^{c\text{-}value}_{ideal}$ (see Figure 5.9), specifically when the performances are assessed at a recall pint (i.e., $NAP_{i=200}$). In the experiments over $\{T\}^{c\text{-}value}_{ideal}$ for $NAP_{i=200}$, encoding information about the order of words in context-windows often worsened the performance of the Euclidean distance and the cosine similarity. However, this information improves the performance of the city block distance. In contrary, in the experiments over $\{T\}^{c\text{-}value}_{Enlarged}$, encoding information about the order of words improves the performance of the Euclidean distance, and exacerbates it for the city block distance. However, for a large recall value (i.e., $NAP_{i=8900}$), the observed results in both $\{T\}^{c\text{-}value}_{ideal}$ and $\{T\}^{c\text{-}value}_{Enlarged}$ are similar in the sense that models that encode this information are not among the top performers.

5.4.3.1 The effect of enlarging the corpus in the presence of invalid terms

The aim is to investigate whether using a larger corpus can enhance the method's performance when the classification is carried out in the presence of invalid terms. Vectors constructed from the set of candidate terms in the $\{T\}^{Y_A T_E A}_{Y_A T_E A}$ are thus augmented by additional co-occurrence frequencies (hereafter, denote by $\{T\}^{Y_A T_E A}_{Enlarged}$). In the enlarged corpus $\{T\}^{Y_A T_E A}_{Enlarged}$, these candidate terms are mentioned more than two million times. Similar to the experiments over $\{T\}^{c\text{-}value}_{Enlarged}$, these mentions of terms are scanned in order to update the co-occurrence frequencies of term vectors. Afterwards, a classification process identical to the one applied to $\{T\}^{Y_A T_E A}_{Y_A T_E A}$ (see Section 5.4.2) is employed to classify the updated vectors obtained from $\{T\}^{Y_A T_E A}_{Enlarged}$.

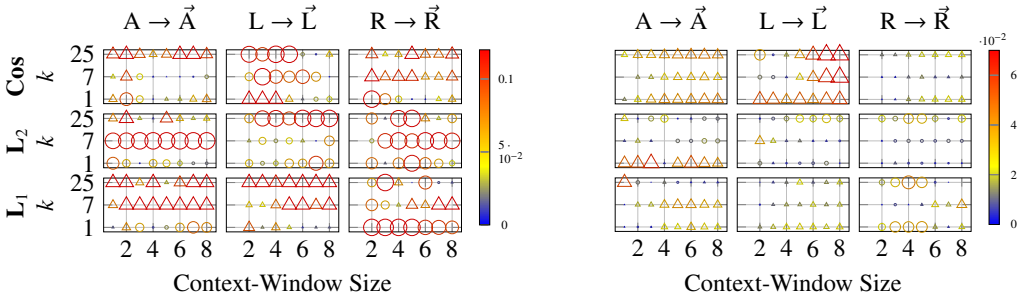
Simialrity Metric	k	A	L	R	$\vec{A}$	$\vec{L}$	$\vec{R}$
Cosine	**1**	-0.62	0.92	-0.32	-0.82	0.74	-0.53
	7	-0.11	0.51	0.39	0.14	-0.04	0.55
	25	-0.19	0.08	0.52	-0.65	0.38	-0.03
Euclid	**1**	0.82	-0.14	0.34	0.9	-0.77	0.36
	7	0.22	0.37	-0.88	0.5	0.24	0.26
	25	0.13	-0.21	-0.06	0.81	0.07	0.85
City block	**1**	0.82	0.87	-0.98	0.88	0.74	-0.95
	7	0.4	0.39	0.63	-0.11	0.05	0.82
	25	0.64	-0.85	-0.39	-0.01	0.25	0.28

(a) Using *NAP* at 2% recall.

Simialrity Metric	k	A	L	R	$\vec{A}$	$\vec{L}$	$\vec{R}$
Cosine	**1**	0.95	0.98	0.92	0.8	0.84	0.01
	7	0.83	0.98	0.85	0.64	0.8	0.66
	25	0.94	0.94	0.93	0.96	0.85	0.84
Euclid	**1**	0.2	0.39	0.18	0.41	0.21	0.3
	7	0.53	0.5	0.25	-0.45	0.78	-0.36
	25	0.33	0.19	-0.02	0.79	0.54	0.97
City block	**1**	0.79	0.17	0.72	0.83	0.78	0.68
	7	0.77	-0.38	0.15	-0.54	0.75	0.12
	25	0.9	-0.25	0.42	0.65	0.76	0.52

(b) Using *NAP* for %100 recall.

Table 5.15: Shown Spearman's correlation coefficient (r_s) between the results obtained in $\{\mathrm{T}\}^{c\text{-}value}_{\mathrm{Enlarged}}$ and $\{\mathrm{T}\}^{c\text{-}value}_{\mathrm{ideal}}$ when the context-window's size considered as the ranking variable and the remaining method's parameters are fixed. Table (a) and (b) shows the observed r_s when performances are computed using *NAP* at 2% and 100% recall, respectively.



(a) $NAP_{i=200}$ (b) $NAP_{i=8900}$

Figure 5.18: The effect of encoding information about the sequential order in the method's performance when dealing with the enlarged corpus. Results are shown in the same format as Figure 5.9. An enhancement in the performance is marked by a circle whereas a decrease is shown by a triangle; the size of the shapes shows the intensity.

Context		$NAP_{i=98}$			$NAP_{i=4278}$		
dir	size	k			k		
		1	7	25	1	7	25
Around	1	0.74(0.83)	0.35(0.72)	0.32(0.65)	0.13(0.36)	0.13(0.4)	0.15(0.4)
	2	0.54(0.69)	0.61(0.81)	0.58(0.77)	0.14(0.36)	0.16(0.4)	0.18(0.4)
	3	0.57(0.72)	0.55(0.67)	0.53(0.73)	0.15(0.36)	0.17(0.38)	0.17(0.38)
	4	0.62(0.78)	0.57(0.72)	0.49(0.71)	0.15(0.36)	0.17(0.37)	0.17(0.37)
	5	0.65(0.78)	0.6(0.68)	0.39(0.62)	0.15(0.35)	0.17(0.37)	0.16(0.37)
	6	0.64(0.77)	0.6(0.69)	0.43(0.58)	0.14(0.35)	0.17(0.36)	0.16(0.36)
	7	0.65(0.78)	0.64(0.74)	0.42(0.56)	0.14(0.35)	0.17(0.36)	0.16(0.36)
	8	0.65(0.78)	0.61(0.72)	0.38(0.54)	0.14(0.36)	0.16(0.36)	0.16(0.36)
Left	1	0.42(0.51)	0.41(0.71)	0.39(0.71)	0.12(0.39)	0.14(0.45)	0.15(0.49)
	2	0.68(0.8)	0.4(0.7)	0.44(0.76)	0.15(0.4)	0.16(0.43)	0.17(0.45)
	3	0.74(0.82)	0.59(0.74)	0.45(0.7)	0.15(0.37)	0.17(0.42)	0.18(0.44)
	4	0.79(0.86)	0.59(0.68)	0.49(0.65)	0.15(0.37)	0.17(0.4)	0.19(0.41)
	5	0.8(0.87)	0.63(0.78)	0.51(0.62)	0.16(0.37)	0.17(0.39)	0.19(0.4)
	6	0.78(0.85)	0.56(0.68)	0.43(0.55)	0.15(0.37)	0.17(0.39)	0.18(0.4)
	7	0.75(0.84)	0.54(0.64)	0.43(0.56)	0.15(0.36)	0.17(0.38)	0.18(0.39)
	8	0.74(0.83)	0.55(0.64)	0.41(0.51)	0.14(0.36)	0.17(0.37)	0.17(0.38)
Right	1	0.48(0.53)	0.24(0.43)	0.27(0.53)	0.1(0.32)	0.11(0.33)	0.11(0.35)
	2	0.45(0.52)	0.32(0.4)	0.32(0.53)	0.1(0.32)	0.11(0.33)	0.12(0.34)
	3	0.47(0.54)	0.35(0.45)	0.31(0.49)	0.1(0.32)	0.12(0.33)	0.12(0.34)
	4	0.53(0.6)	0.34(0.44)	0.34(0.54)	0.11(0.33)	0.12(0.33)	0.12(0.34)
	5	0.54(0.62)	0.48(0.56)	0.37(0.55)	0.11(0.33)	0.13(0.34)	0.13(0.35)
	6	0.56(0.64)	0.43(0.55)	0.47(0.6)	0.12(0.34)	0.13(0.34)	0.13(0.35)
	7	0.58(0.66)	0.38(0.53)	0.47(0.6)	0.12(0.34)	0.13(0.34)	0.13(0.35)
	8	0.59(0.67)	0.42(0.54)	0.38(0.53)	0.12(0.34)	0.13(0.34)	0.13(0.34)
Baseline		**0.273(0.87)**			**0.12(0.5)**		

(a) Sequential Order of Words Discarded

$NAP_{i=98}$			$NAP_{i=4278}$		
k			k		
1	7	25	1	7	25
0.49(0.55)	0.21(0.42)	0.27(0.58)	0.1(0.32)	0.11(0.35)	0.12(0.37)
0.49(0.64)	0.2(0.4)	0.3(0.6)	0.1(0.33)	0.11(0.34)	0.13(0.37)
0.49(0.62)	0.21(0.39)	0.3(0.6)	0.1(0.32)	0.11(0.34)	0.13(0.36)
0.48(0.6)	0.26(0.41)	0.32(0.61)	0.09(0.32)	0.11(0.34)	0.13(0.35)
0.47(0.58)	0.29(0.43)	0.34(0.64)	0.09(0.32)	0.11(0.33)	0.13(0.35)
0.46(0.57)	0.32(0.46)	0.33(0.63)	0.1(0.32)	0.11(0.33)	0.12(0.35)
0.46(0.57)	0.28(0.45)	0.32(0.61)	0.09(0.32)	0.11(0.33)	0.12(0.35)
0.46(0.57)	0.27(0.44)	0.36(0.63)	0.1(0.32)	0.11(0.33)	0.12(0.35)
0.42(0.51)	0.41(0.71)	0.39(0.71)	0.12(0.39)	0.14(0.45)	0.15(0.49)
0.66(0.77)	0.47(0.71)	0.48(0.76)	0.13(0.37)	0.14(0.42)	0.13(0.45)
0.66(0.78)	0.46(0.7)	0.47(0.76)	0.12(0.36)	0.14(0.4)	0.12(0.43)
0.67(0.78)	0.54(0.7)	0.45(0.74)	0.12(0.36)	0.15(0.4)	0.12(0.42)
0.67(0.78)	0.47(0.62)	0.42(0.68)	0.13(0.38)	0.15(0.4)	0.12(0.42)
0.68(0.78)	0.44(0.6)	0.42(0.65)	0.13(0.36)	0.13(0.38)	0.12(0.41)
0.68(0.78)	0.38(0.57)	0.4(0.63)	0.12(0.35)	0.12(0.37)	0.12(0.41)
0.63(0.76)	0.44(0.64)	0.36(0.61)	0.11(0.35)	0.12(0.37)	0.11(0.4)
0.48(0.53)	0.24(0.43)	0.27(0.53)	0.1(0.32)	0.11(0.33)	0.11(0.35)
0.56(0.71)	0.21(0.43)	0.3(0.57)	0.11(0.32)	0.11(0.33)	0.12(0.35)
0.54(0.69)	0.22(0.4)	0.32(0.62)	0.1(0.32)	0.11(0.33)	0.12(0.35)
0.53(0.65)	0.21(0.35)	0.33(0.59)	0.1(0.32)	0.11(0.33)	0.12(0.34)
0.51(0.62)	0.26(0.39)	0.3(0.62)	0.1(0.32)	0.11(0.33)	0.12(0.34)
0.5(0.61)	0.32(0.45)	0.32(0.64)	0.1(0.32)	0.11(0.33)	0.12(0.34)
0.49(0.6)	0.34(0.47)	0.33(0.63)	0.09(0.32)	0.11(0.33)	0.12(0.34)
0.48(0.6)	0.36(0.46)	0.31(0.65)	0.09(0.32)	0.11(0.32)	0.12(0.34)
0.273(0.87)			**0.12(0.5)**		

(b) Sequential Order of Words Encoded

Table 5.16: The performances observed over the $\{T\}^{Y_A T_E A}_{\text{Enlarged}}$ when $|R_s| = 100$ and similarities are computed using the *cosine* between vectors. Similar to the previous experiments, the performance is reported with respect to the observed NAP_i, for $i = 98$ (i.e., recall = 0.02) and $i = 4278$ (i.e., recall 100%). Numbers placed in parentheses show NAP_i when the precision is computed with respect to the number of valid terms.

The method's performance over $\{T\}^{Y_A T_E A}_{\text{Enlarged}}$ is reported in Tables 5.16, 5.17, and 5.18. These numbers are plotted in Figure 5.19. To study the effect of enlarging the corpus, these results are compared with their corresponding values obtained from the earlier experiment over $\{T\}^{Y_A T_E A}_{Y_A T_E A}$ (i.e., results reported in Section 5.4.2; see Figure 5.11). Here, enlarging the corpus size marginally enhances the best observed performance. Particularly, although enlarging the corpus enhances the discriminatory power of the models, it does not necessarily improve the method's ability to filter invalid terms (see *NAP* for valid terms reported in Tables 5.16, 5.17, and 5.18; that is, numbers placed in parentheses).

Figure 5.20 plots changes in the method's performance caused by enlarging the corpus at 100% recall (i.e., $NAP_{i=4278}$ in this experiment). Compared to experiments over $\{T\}^{\text{c-}value}_{\text{Enlarged}}$ (see Figure 5.15), increasing the size of corpus in $\{T\}^{Y_A T_E A}_{Y_A T_E A}$ enhances the performance disregarding the method's parameters. However, even with these enhancements, performances remain below the baseline for many combinations of parameters, particularly when similarities are computed using the Euclidean and the city block distance. Similarly, Figure 5.21 shows how enlarging the corpus effects the observed performances at a small recall point such as $NAP_{i=98}$ (i.e., recall %2). As shown, compared to the experiments over $\{T\}^{\text{c-}value}_{\text{Enlarged}}$ (see Figure 5.16), enlarging the corpus has a more steady positive effect on the discriminatory power of the constructed models when the classification task

Context		$NAP_{i=98}$			$NAP_{i=4278}$		
dir	size	k			k		
		1	7	25	1	7	25
Around	1	0.53(0.74)	0.48(0.66)	0.48(0.58)	0.13(0.43)	0.12(0.48)	0.13(0.37)
	2	0.54(0.77)	0.47(0.66)	0.43(0.72)	0.12(0.46)	0.12(0.46)	0.1(0.45)
	3	0.49(0.73)	0.5(0.72)	0.52(0.73)	0.11(0.44)	0.11(0.44)	0.11(0.44)
	4	0.47(0.7)	0.56(0.75)	0.66(0.81)	0.13(0.36)	0.11(0.44)	0.12(0.37)
	5	0.43(0.67)	0.56(0.72)	0.66(0.75)	0.12(0.42)	0.11(0.43)	0.11(0.42)
	6	0.42(0.64)	0.57(0.74)	0.46(0.65)	0.11(0.41)	0.11(0.43)	0.1(0.41)
	7	0.44(0.64)	0.55(0.68)	0.42(0.63)	0.11(0.41)	0.11(0.43)	0.1(0.41)
	8	0.44(0.63)	0.53(0.71)	0.4(0.6)	0.11(0.41)	0.11(0.43)	0.1(0.41)
Left	1	0.64(0.79)	0.54(0.79)	0.61(0.82)	0.13(0.5)	0.13(0.49)	0.13(0.51)
	2	0.63(0.81)	0.58(0.8)	0.58(0.77)	0.13(0.48)	0.14(0.48)	0.13(0.49)
	3	0.62(0.76)	0.61(0.79)	0.57(0.78)	0.13(0.46)	0.13(0.46)	0.13(0.47)
	4	0.6(0.75)	0.54(0.75)	0.57(0.77)	0.12(0.46)	0.13(0.45)	0.12(0.46)
	5	0.6(0.75)	0.59(0.78)	0.53(0.75)	0.12(0.45)	0.12(0.45)	0.12(0.45)
	6	0.56(0.74)	0.59(0.78)	0.58(0.78)	0.12(0.41)	0.12(0.44)	0.12(0.42)
	7	0.55(0.74)	0.59(0.79)	0.6(0.75)	0.12(0.44)	0.12(0.44)	0.12(0.44)
	8	0.56(0.7)	0.54(0.73)	0.53(0.72)	0.11(0.44)	0.11(0.44)	0.11(0.44)
Right	1	0.37(0.67)	0.5(0.74)	0.57(0.74)	0.09(0.44)	0.1(0.43)	0.1(0.44)
	2	0.39(0.71)	0.51(0.69)	0.32(0.69)	0.1(0.44)	0.1(0.43)	0.09(0.44)
	3	0.36(0.69)	0.51(0.75)	0.32(0.68)	0.1(0.44)	0.1(0.43)	0.09(0.43)
	4	0.38(0.7)	0.55(0.74)	0.34(0.69)	0.1(0.43)	0.1(0.42)	0.09(0.43)
	5	0.38(0.68)	0.58(0.76)	0.36(0.69)	0.1(0.43)	0.11(0.42)	0.09(0.43)
	6	0.38(0.68)	0.57(0.7)	0.51(0.72)	0.11(0.42)	0.11(0.4)	0.1(0.42)
	7	0.38(0.67)	0.51(0.67)	0.53(0.69)	0.11(0.36)	0.12(0.35)	0.11(0.4)
	8	0.39(0.67)	0.49(0.67)	0.49(0.69)	0.1(0.36)	0.1(0.42)	0.09(0.35)
Baseline		**0.273(0.87)**			**0.12(0.5)**		

(a) Sequential Order of Words Discarded

$NAP_{i=98}$			$NAP_{i=4278}$		
k			k		
1	7	25	1	7	25
0.37(0.69)	0.5(0.76)	0.58(0.76)	0.1(0.45)	0.1(0.45)	0.1(0.45)
0.4(0.69)	0.5(0.77)	0.58(0.76)	0.1(0.44)	0.1(0.44)	0.1(0.45)
0.43(0.68)	0.53(0.78)	0.58(0.76)	0.1(0.44)	0.11(0.44)	0.11(0.44)
0.48(0.72)	0.53(0.76)	0.6(0.75)	0.11(0.44)	0.11(0.43)	0.11(0.44)
0.48(0.71)	0.52(0.76)	0.59(0.75)	0.11(0.43)	0.11(0.43)	0.12(0.44)
0.52(0.7)	0.59(0.76)	0.59(0.76)	0.11(0.34)	0.11(0.43)	0.11(0.34)
0.55(0.72)	0.63(0.75)	0.59(0.76)	0.12(0.37)	0.11(0.43)	0.13(0.36)
0.57(0.75)	0.67(0.74)	0.6(0.76)	0.12(0.39)	0.11(0.42)	0.13(0.38)
0.64(0.79)	0.54(0.79)	0.61(0.82)	0.13(0.5)	0.13(0.49)	0.13(0.51)
0.58(0.71)	0.67(0.78)	0.39(0.7)	0.11(0.37)	0.11(0.33)	0.1(0.35)
0.59(0.78)	0.59(0.79)	0.59(0.71)	0.12(0.41)	0.13(0.39)	0.11(0.38)
0.56(0.75)	0.64(0.75)	0.65(0.77)	0.13(0.43)	0.14(0.42)	0.13(0.41)
0.57(0.72)	0.69(0.76)	0.69(0.82)	0.13(0.42)	0.13(0.41)	0.13(0.42)
0.53(0.68)	0.66(0.73)	0.67(0.8)	0.13(0.4)	0.12(0.39)	0.12(0.42)
0.58(0.7)	0.51(0.58)	0.6(0.67)	0.12(0.42)	0.12(0.41)	0.12(0.41)
0.48(0.69)	0.51(0.58)	0.57(0.69)	0.11(0.42)	0.12(0.41)	0.11(0.41)
0.37(0.67)	0.5(0.74)	0.57(0.74)	0.09(0.44)	0.1(0.43)	0.1(0.44)
0.39(0.68)	0.53(0.77)	0.58(0.76)	0.1(0.44)	0.1(0.44)	0.1(0.45)
0.42(0.71)	0.54(0.77)	0.59(0.76)	0.1(0.44)	0.1(0.44)	0.11(0.44)
0.44(0.71)	0.54(0.77)	0.59(0.75)	0.11(0.44)	0.11(0.44)	0.11(0.44)
0.47(0.73)	0.52(0.76)	0.6(0.76)	0.11(0.44)	0.11(0.43)	0.11(0.44)
0.49(0.73)	0.53(0.77)	0.6(0.76)	0.11(0.43)	0.11(0.43)	0.11(0.44)
0.52(0.75)	0.56(0.77)	0.62(0.77)	0.11(0.43)	0.11(0.43)	0.11(0.44)
0.52(0.75)	0.56(0.76)	0.61(0.76)	0.11(0.42)	0.11(0.43)	0.12(0.42)
0.27(0.87)			**0.12(0.5)**		

(b) Sequential Order of Words Encoded

Table 5.17: The results observed in $\{\mathrm{T}\}^{\mathrm{YATEA}}_{\mathrm{Enlarged}}$ when $|R_s| = 100$ and similarities are computed using the *Euclidean* distance.

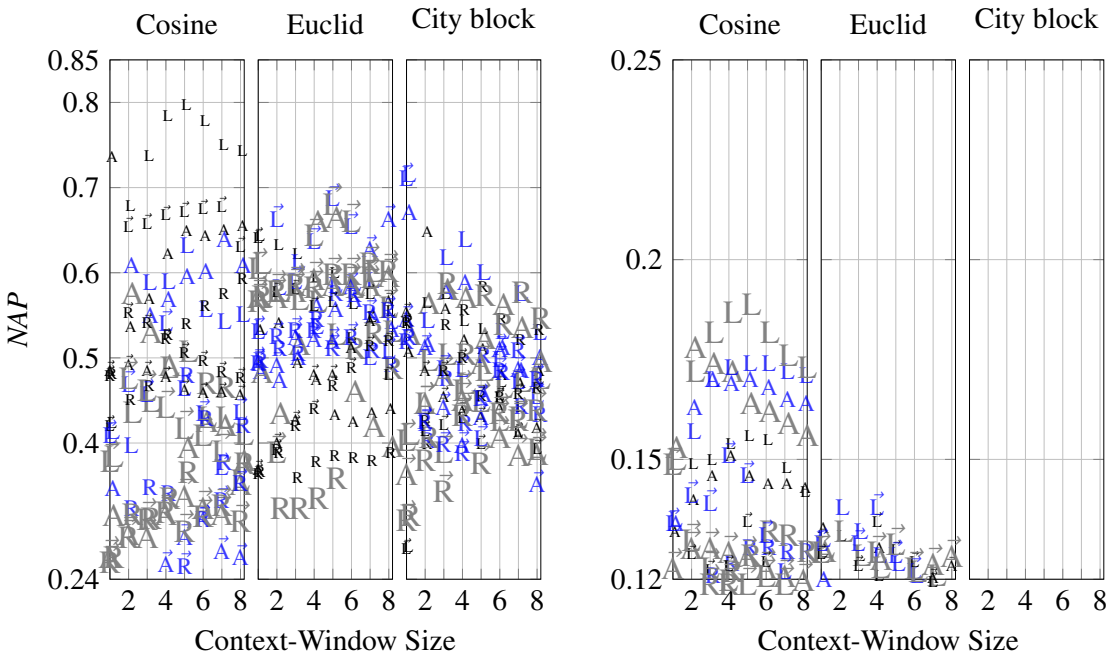
Context		$NAP_{i=98}$			$NAP_{i=4278}$		
dir	size	k			k		
		1	7	25	1	7	25
Around	1	0.51(0.69)	0.67(0.8)	0.53(0.74)	0.1(0.36)	0.1(0.34)	0.09(0.35)
	2	0.65(0.77)	0.52(0.67)	0.56(0.73)	0.1(0.36)	0.09(0.31)	0.1(0.35)
	3	0.56(0.69)	0.58(0.71)	0.58(0.72)	0.1(0.36)	0.1(0.32)	0.1(0.35)
	4	0.52(0.7)	0.59(0.7)	0.57(0.7)	0.1(0.37)	0.1(0.34)	0.1(0.36)
	5	0.51(0.67)	0.46(0.59)	0.52(0.68)	0.1(0.37)	0.1(0.35)	0.1(0.36)
	6	0.48(0.66)	0.51(0.6)	0.56(0.71)	0.1(0.35)	0.1(0.36)	0.1(0.33)
	7	0.47(0.64)	0.49(0.61)	0.54(0.71)	0.11(0.39)	0.1(0.35)	0.11(0.38)
	8	0.49(0.68)	0.5(0.61)	0.5(0.7)	0.11(0.38)	0.1(0.36)	0.11(0.36)
Left	1	0.28(0.21)	0.72(0.85)	0.41(0.72)	0.06(0.26)	0.09(0.32)	0.07(0.29)
	2	0.57(0.71)	0.55(0.74)	0.5(0.71)	0.1(0.36)	0.1(0.35)	0.09(0.32)
	3	0.58(0.73)	0.62(0.73)	0.5(0.7)	0.1(0.35)	0.11(0.33)	0.09(0.32)
	4	0.54(0.71)	0.64(0.74)	0.47(0.7)	0.1(0.35)	0.11(0.33)	0.09(0.33)
	5	0.53(0.7)	0.6(0.73)	0.49(0.68)	0.1(0.35)	0.11(0.34)	0.1(0.34)
	6	0.47(0.64)	0.53(0.7)	0.47(0.64)	0.1(0.34)	0.1(0.33)	0.09(0.34)
	7	0.44(0.63)	0.58(0.7)	0.47(0.64)	0.1(0.36)	0.1(0.33)	0.1(0.35)
	8	0.48(0.67)	0.53(0.64)	0.48(0.62)	0.1(0.36)	0.1(0.32)	0.1(0.35)
Right	1	0.54(0.7)	0.52(0.65)	0.32(0.57)	0.09(0.35)	0.09(0.35)	0.07(0.33)
	2	0.4(0.54)	0.4(0.51)	0.41(0.54)	0.08(0.33)	0.08(0.33)	0.08(0.31)
	3	0.54(0.64)	0.48(0.58)	0.59(0.65)	0.1(0.34)	0.09(0.33)	0.09(0.32)
	4	0.56(0.71)	0.42(0.6)	0.51(0.71)	0.1(0.33)	0.1(0.33)	0.09(0.31)
	5	0.58(0.73)	0.5(0.63)	0.57(0.65)	0.1(0.33)	0.1(0.33)	0.09(0.31)
	6	0.55(0.71)	0.48(0.63)	0.54(0.64)	0.1(0.33)	0.09(0.32)	0.09(0.31)
	7	0.52(0.7)	0.44(0.62)	0.58(0.69)	0.1(0.33)	0.09(0.33)	0.09(0.32)
	8	0.53(0.71)	0.43(0.61)	0.55(0.66)	0.1(0.34)	0.1(0.34)	0.09(0.32)
Baseline		**0.273(0.87)**			**0.12(0.5)**		

(a) Sequential Order of Words Discarded

$NAP_{i=98}$			$NAP_{i=4278}$		
k			k		
1	7	25	1	7	25
0.56(0.77)	0.55(0.7)	0.37(0.7)	0.1(0.37)	0.08(0.32)	0.08(0.36)
0.49(0.71)	0.52(0.67)	0.45(0.73)	0.09(0.34)	0.09(0.32)	0.09(0.33)
0.46(0.7)	0.45(0.54)	0.49(0.71)	0.09(0.33)	0.09(0.32)	0.09(0.31)
0.44(0.61)	0.4(0.55)	0.46(0.64)	0.09(0.32)	0.09(0.31)	0.09(0.31)
0.43(0.65)	0.43(0.62)	0.5(0.66)	0.09(0.31)	0.09(0.31)	0.09(0.3)
0.45(0.65)	0.51(0.66)	0.41(0.6)	0.08(0.3)	0.09(0.31)	0.08(0.3)
0.41(0.63)	0.49(0.66)	0.39(0.58)	0.08(0.31)	0.09(0.32)	0.08(0.31)
0.42(0.59)	0.36(0.53)	0.39(0.55)	0.08(0.31)	0.08(0.31)	0.08(0.3)
0.28(0.21)	0.72(0.85)	0.41(0.72)	0.06(0.26)	0.09(0.32)	0.07(0.29)
0.42(0.64)	0.43(0.6)	0.45(0.67)	0.09(0.35)	0.09(0.29)	0.09(0.35)
0.42(0.66)	0.49(0.73)	0.4(0.51)	0.09(0.31)	0.09(0.3)	0.08(0.3)
0.43(0.69)	0.49(0.67)	0.47(0.68)	0.1(0.33)	0.09(0.33)	0.09(0.31)
0.4(0.59)	0.41(0.51)	0.46(0.68)	0.09(0.31)	0.08(0.3)	0.09(0.3)
0.46(0.68)	0.5(0.61)	0.43(0.66)	0.09(0.31)	0.09(0.3)	0.09(0.31)
0.42(0.68)	0.48(0.65)	0.44(0.69)	0.09(0.32)	0.09(0.31)	0.09(0.32)
0.4(0.62)	0.46(0.64)	0.4(0.66)	0.09(0.32)	0.09(0.31)	0.09(0.32)
0.54(0.7)	0.52(0.65)	0.32(0.57)	0.09(0.35)	0.09(0.35)	0.07(0.33)
0.43(0.66)	0.43(0.62)	0.39(0.51)	0.09(0.37)	0.09(0.38)	0.08(0.3)
0.49(0.64)	0.4(0.62)	0.35(0.46)	0.08(0.3)	0.08(0.31)	0.08(0.3)
0.5(0.63)	0.39(0.61)	0.44(0.5)	0.09(0.32)	0.09(0.34)	0.08(0.3)
0.46(0.6)	0.46(0.61)	0.38(0.59)	0.09(0.35)	0.09(0.34)	0.09(0.32)
0.46(0.62)	0.49(0.64)	0.44(0.58)	0.08(0.31)	0.09(0.31)	0.08(0.31)
0.46(0.62)	0.51(0.63)	0.43(0.57)	0.08(0.31)	0.08(0.3)	0.08(0.3)
0.47(0.62)	0.48(0.6)	0.44(0.6)	0.09(0.31)	0.09(0.31)	0.08(0.31)
0.27(0.87)			**0.12(0.5)**		

(b) Sequential Order of Words Encoded

Table 5.18: The results obtained in $\{\mathrm{T}\}^{\mathrm{YATEA}}_{\mathrm{Enlarged}}$ when $|R_s| = 100$ and similarities are computed using the *city block* distance.



(a) $NAP_{i=98}$ (b) $NAP_{i=4278}$

Figure 5.19: The NAP_i observed in $\{\mathrm{T}\}^{\mathrm{YATEA}}_{\mathrm{Enlarged}}$ for $i = 98$ (i.e., 2% recall) and $i = 4278$ (i.e., recall 100%) are shown in (a) and (b), respectively.

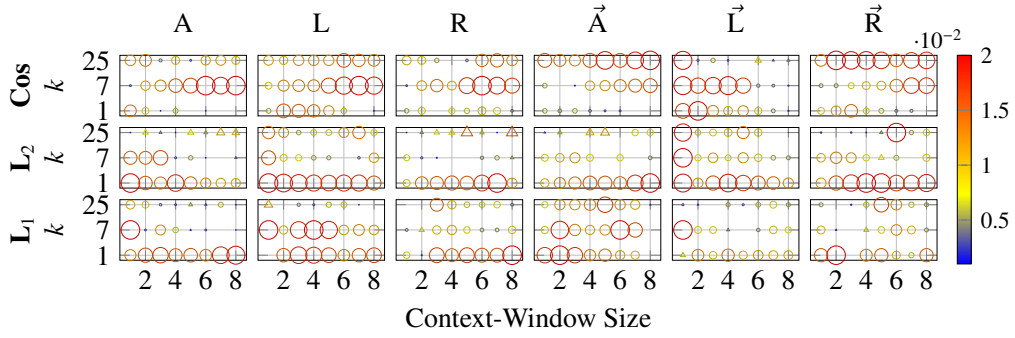


Figure 5.20: Changes in the performance of the method caused by increasing the size of the input corpus when the performance is measured using *NAP* for 100% recall in the experiments over $\{\mathrm{T}\}^{\mathrm{YATEA}}_{\mathrm{Enlarged}}$. Shown is the absolute value of the difference between the performance obtained from a model constructed in $\{\mathrm{T}\}^{\mathrm{YATEA}}_{\mathrm{YATEA}}$ and the corresponding model in $\{\mathrm{T}\}^{\mathrm{YATEA}}_{\mathrm{Enlarged}}$. Triangles denote negative change whereas circles show positive change. The size/colour of shapes represents the amount of changes. The *x*-axis shows various configurations of context-windows. The *y*-axis represents classification parameters.

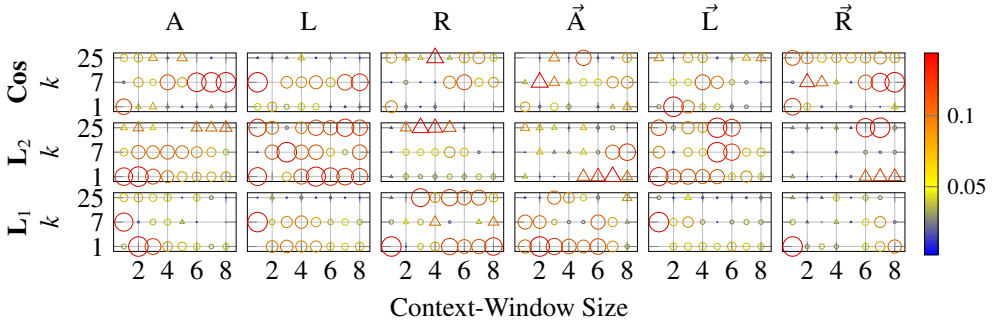


Figure 5.21: The changes in the performance caused by increasing the size of the input corpus when the method's performance is measured using $NAP_{i=200}$ (i.e, 2% recall) over $\{T\}^{Y_A T_E A}_{\text{Enlarged}}$. The presentation format is similar to Figure 5.20: circles show positive effects whereas triangles show negative impacts.

is accomplished for a set of candidate terms that contain invalid terms.

With an exception to the size of context-windows, parameters that gives in the best performance in $\{T\}^{Y_A T_E A}_{Y_A T_E A}$ also results the best performance in $\{T\}^{Y_A T_E A}_{\text{Enlarged}}$. In both $\{T\}^{Y_A T_E A}_{Y_A T_E A}$ and $\{T\}^{Y_A T_E A}_{\text{Enlarged}}$, the most discriminative models are built using context-windows that extend to the left side of candidate terms. In contrast to experiments over $\{T\}^{Y_A T_E A}_{Y_A T_E A}$, in the experiments over $\{T\}^{Y_A T_E A}_{\text{Enlarged}}$ extending context-windows more than 5 tokens often diminishes the performance. The cosine metric on average shows the best performance; in this case, a small k results in the best performance at small recall points whereas large k must be chosen at large recall points (e.g., $k = 1$ at 2% recall and $k = 25$ at 100% recall, respectively).

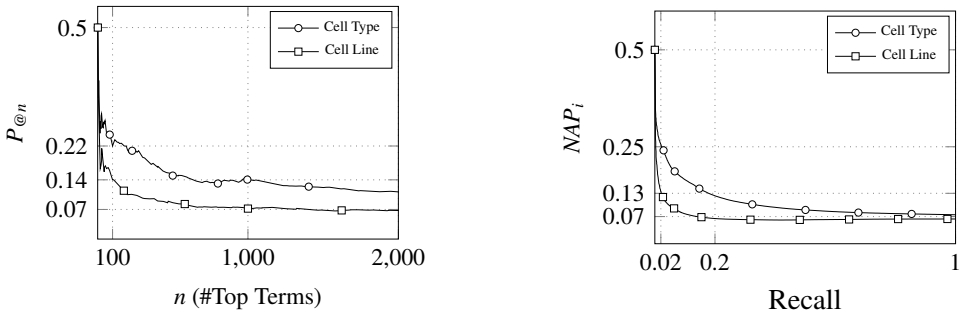
5.4.4 Evaluating Parameters Across Concept Categories

Terms are often classified in several categories of concepts; therefore, the identification of co-hyponym terms can go beyond one concept category. For instance, in the domain of *molecular biology* (and accordingly in the GENIA corpus), several categories of terms (e.g., *cell line*, *cell type*, etc.) other than *protein* are conceived. The question here is that whether the same configuration of the context-window and the classification's parameters can be used for identify terms from different concept categories. That is to say, if a model shows the best performance for identifying a category of terms such as *protein*, would it be also the top performer for extracting terms that belong to other categories such as , *cell line*, and *cell type*?

To answer the questions asked above, the reported evaluation in the Section 5.4.1 over $\{T\}^{c\text{-}value}_{\text{ideal}}$ are repeated; however, for identifying terms that are classified under the concept category of *cell type* and *cell line* (i.e., terms that are annotated as `G#cell_type` and `G#cell_line` in the GENIA corpus, respectively). Table 5.19 shows statistics for these two categories of term in the corpus. Similar to the description given in Section 5.3 for protein terms, terms that are annotated at least once as cell type or cell line are collected from the corpus. Those terms that are annotated in one additional category are marked as

Category	Frequency (mentions)	#Distinct Entry	#Polysemous Entry
Cell Type	8,257	2,097	178
Cell Line	5,944	2,261	154

Table 5.19: Shown are the statistics of the co-hyponym terms in the two categories of *cell type* and *virus* in the GENIA corpus. Polysemous entries are subset of distinct entries.



(a) (b)

Figure 5.22: Baseline performances when extracting terms from the categories of *cell type* and *cell line* in $\{T\}^{c\text{-}value}_{ideal}$ ranked terms: (a) shows the proportion of terms in these categories in the top 2000 entries in $\{T\}^{c\text{-}value}_{ideal}$ (i.e., $P_{@n}$ for $1 \leq n \leq 2000$); (b) shows the performance computed by NAP_i for the complete range of recall values—that is, $i = 2,097$ and $i = 2261$ for terms in the category of *cell type* and *cell line*, respectively.

polysemous.

Accordingly, when extracting terms that belong to the concept category of cell type and cell line, the random baseline approaches to $\frac{2097}{34077} = 0.061$ and $\frac{2261}{34077} = 0.066$, respectively. Figure 5.22 also shows the c-*value* ranking baselines (i.e., the baselines computed using the set of ranked terms in $\{T\}^{c\text{-}value}_{ideal}$). As shown, at the small recall point of 2% (i.e., $NAP_{i=42}$ for *cell type* and $NAP_{i=45}$ for *cell line*), the c-*value* ranking baseline is 0.255 and 0.134 for the cell type and cell line categories, respectively. However, at 100% recall (i.e., *NAP* at $i = 2097$ and $i = 2261$ for terms in the category of cell type and cell line, respectively), the performance of the c-*value* baseline is similar to the random baseline. Since terms that belong to the category of cell line are less frequent than cell type, they are given lower ranks by the c-*value* measure. As a result, although the number of distinct terms annotated as cell line is larger than cell type, the computed c-*value* baseline for cell line is less than cell type.

A classification process identical to the one employed for identifying protein terms in Section 5.4.1 is carried out for extracting terms that belong to the category of cell type and cell line. As shown in Figure 5.22a, the employed R_s (i.e., the top 100 c-*value* ranked terms) contains 14 terms from the cell line category and 22 terms from the cell type category. The obtained results are plotted in Figures 5.23 and 5.24.

As an initial inspection of the results shows, answering the questions asked above is not straight forward, particularly, at small recall points. Assuming that the method's

parameters are fixed, then the performance appears to be sensitive to the chosen targeted category of concepts. That is to say, to obtain the best performances for identifying each category of co-hyponyms terms, often context-windows must be reconfigured with respect to the evaluated parameters. For instance, at 2% recall, if similarities are computed using the cosine measure, then context-windows that extend *around* the terms shows the best performance for identifying *cell type* terms (Figure 5.23a). However, under the similar conditions, context-windows that extend to the *left* side of terms shows the best performance for identifying *cell line* terms (Figure 5.24a).

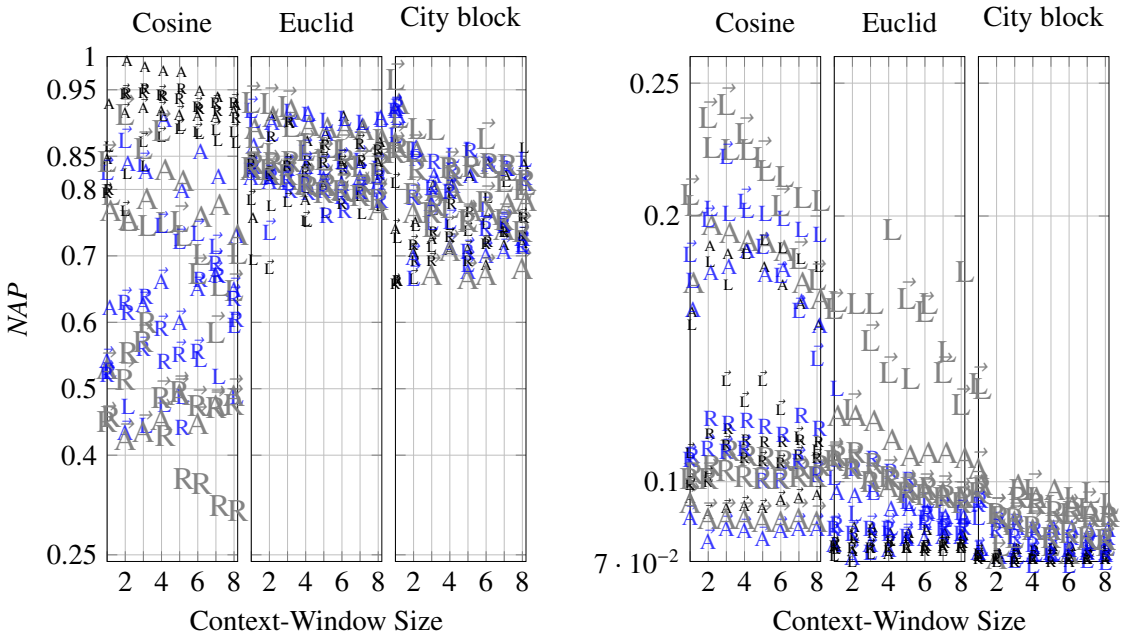
When it comes to the choice of choice of k in the classification process, a similar conclusion as to the parameters of context-windows can be drawn, too. For different categories of concepts, the best performances are obtained using different values of k (e.g., $k = 1$ in Figure 5.23a vs. $= 7$ in Figure 5.24a). However, concerning the choice of similarity metric, the observations are predominantly comparable across categories of concepts. Except for the small recall values, the cosine measure outperforms the other evaluated metrics (see discussions related to Figures 5.10, 5.12, and 5.17).

The experiments are also repeated over the enlarged corpus—that is, $\{\mathrm{T}\}^{\textit{c-value}}_{\mathrm{Enlarged}}$. The obtained performances are abridged in Figure 5.25 and 5.26, which also corroborate the conclusions drawn above.

A comparison between the results that are plotted earlier in Figures 5.23 and 5.24, and the results reported in Figures 5.25 and 5.26 (i.e., comparing the method's performance in $\{\mathrm{T}\}^{\textit{c-value}}_{\mathrm{ideal}}$ and $\{\mathrm{T}\}^{\textit{c-value}}_{\mathrm{Enlarged}}$) leads to a discussion similar to the one proposed in Section 5.4.3: enlarging the corpus does not necessarily enhance the observed performances. Taking the results reported throughout this section into consideration, it becomes evident that the effect of enlarging the corpus not only depends on the configuration of context-windows and the chosen values for the classification's parameters (as suggested in Section 5.4.3), but also on the targeted category of concepts. For instance, when the method's performance is investigated at 100% recall and the cosine measure is employed to compute similarities, enlarging the corpus has a positive effect on the performance when extracting terms that belong to the category of cell type (compare Figures 5.23b and 5.25b). However, under the same conditions, the result is the opposite when extracting cell line terms—that is, a decrease in the performance is observed (compare the cosine section of Figures 5.24b and 5.26b).

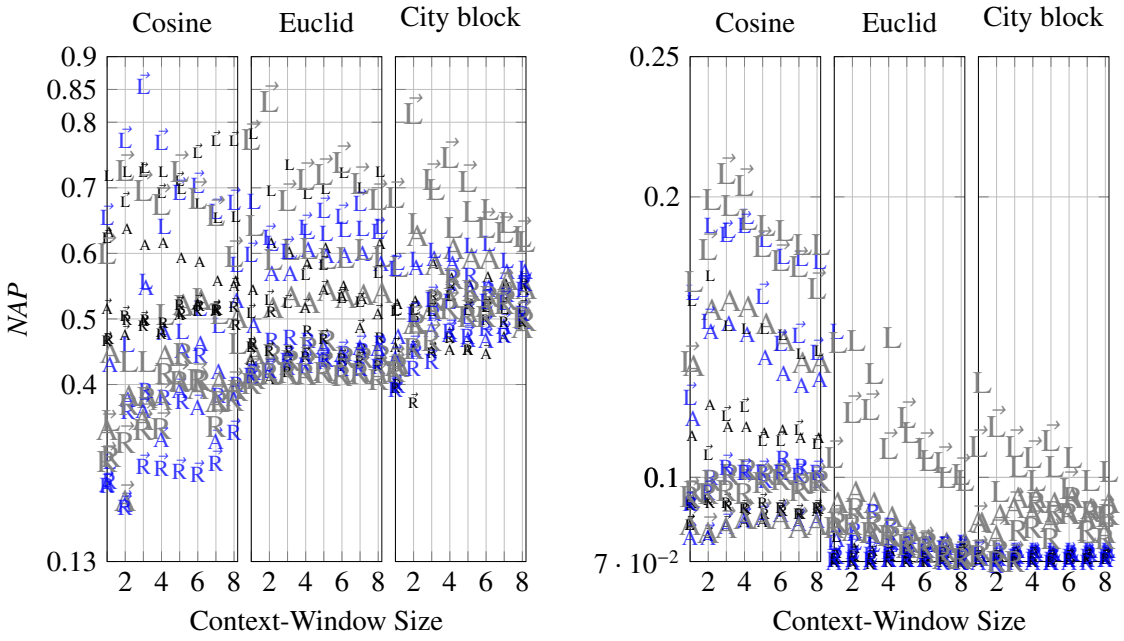
5.4.5 Averaging Performances Across Concept Categories

The construction of a vector space model and configuring it for a particular category of concepts would result in the best possible performance, as shown in the previous section. However, this practice cannot be feasible for a few reasons. The construction of a model, even with a reduced dimensionality, demands computational resources that may not be available in order to construct a model for category of concepts. It is therefore likely that a single model is employed to identify a variety of co-hyponymy relationships in an application. In the context of this chapter, for example, a single model could be used to identify terms from the categories of protein, cell type, and cell line. One way to choose



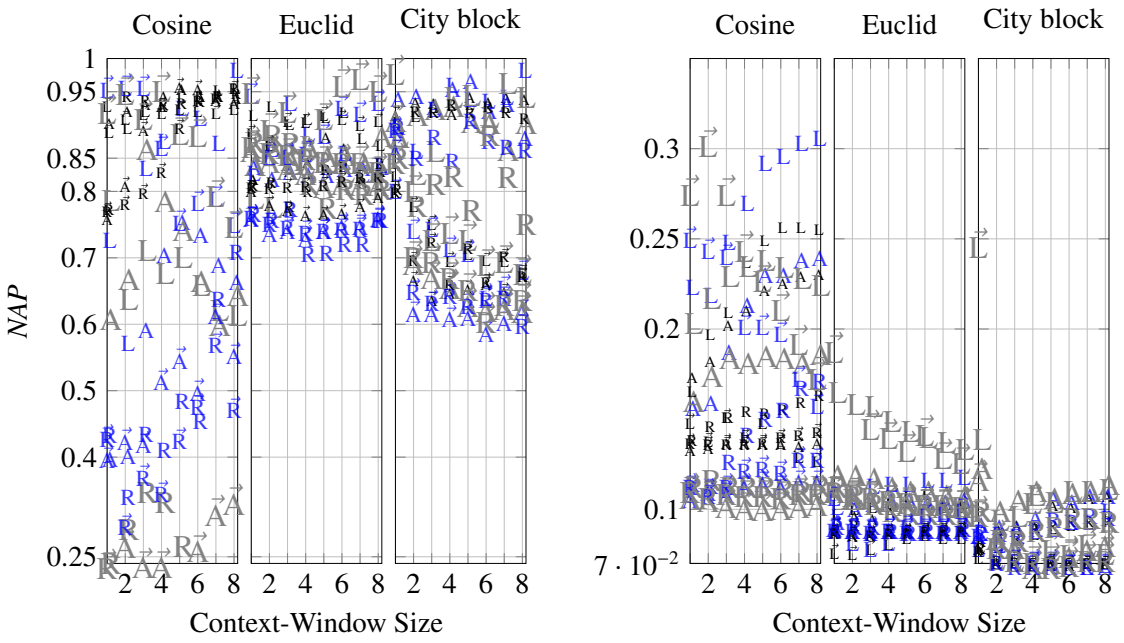
(a) $\boldsymbol{NAP_{i=42}}$ (b) $\boldsymbol{NAP_{i=2097}}$

Figure 5.23: Performance over$\{\mathrm{T}\}^{\text{c-}value}_{\text{ideal}}$ for extracting terms from the *cell type* category. The notation is similar to previous figures: letters show the direction in which context-windows are extended; their size/colour denote the value of k, and the $\vec{\square}$ implies encoding information about words order. The y-axis's minimum value shows the c-*value* baseline.



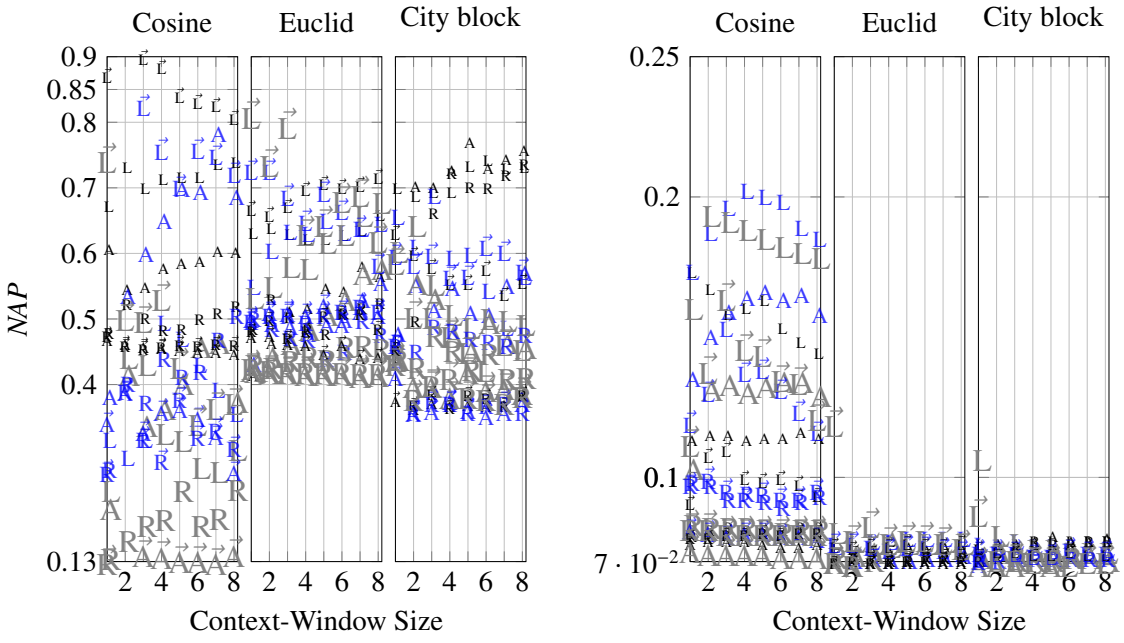
(a) $\boldsymbol{NAP_{i=45}}$ (b) $\boldsymbol{NAP_{i=2261}}$

Figure 5.24: Performance over $\{\mathrm{T}\}^{\text{c-}value}_{\text{ideal}}$ for extracting terms from the category of *cell line*. The y-axis's minimum value shows the c-*value* baseline.



(a) $\boldsymbol{NAP_{i=42}}$ (b) $\boldsymbol{NAP_{i=2097}}$

Figure 5.25: The method's performance over $\{\mathrm{T}\}^{\textit{c-value}}_{\mathrm{Enlarged}}$ for extracting terms in the category of *cell type*.



(a) $\boldsymbol{NAP_{i=45}}$ (b) $\boldsymbol{NAP_{i=2261}}$

Figure 5.26: The method's performance over $\{\mathrm{T}\}^{\textit{c-value}}_{\mathrm{Enlarged}}$ for identifying terms in the category of *cell line*.

a configuration for this model is to use the average of performances across the categories.

In this section, the average performance of the method across the concept categories of protein, cell type, and cell line are reported when the parameters of the context-window and the classification process are set differently. To do so, similar to the evaluation of an information retrieval system in a task that involves a set of queries (e.g., as suggested by Manning et al., 2009, chap. 8), the average of the arithmetic mean of recorded non-interpolated average precisions (i.e., NAP_i) is employed as a single-figure measure of the method's performance across categories of concepts. This arithmetic mean average of performances (*MAP*) is simply the sum of the observed NAP_i for the three aforementioned categories of terms divided by the number of categories (i.e., 3 in here).

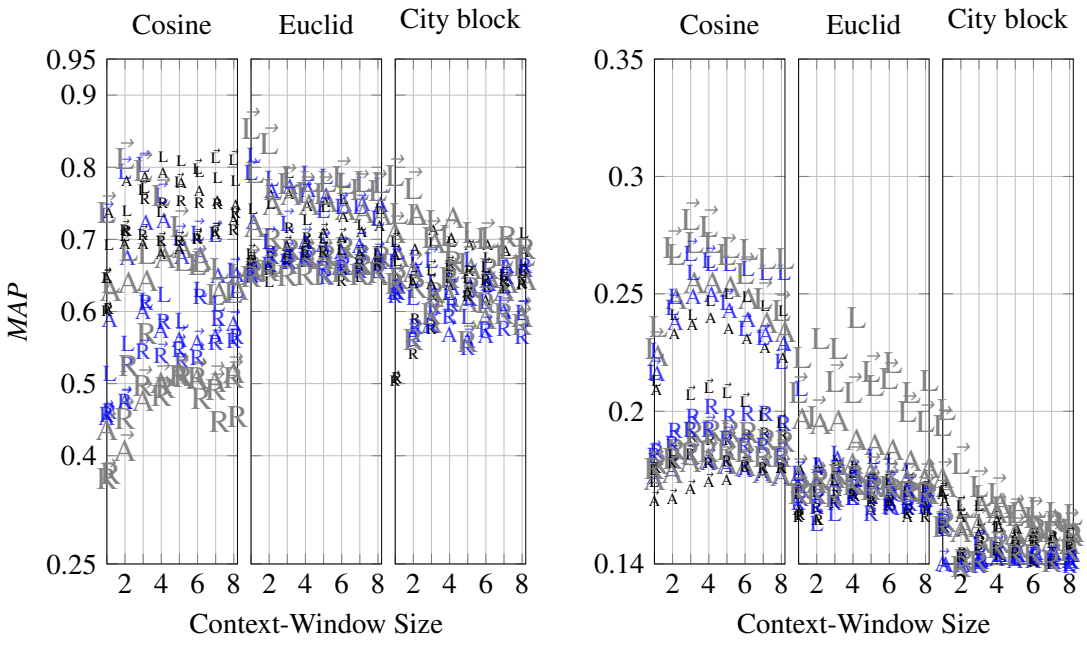
For each of the evaluated datasets, the observed *MAP* is reported for the two recall points of 2% and 100%. Figures 5.27 and 5.28 show the results over the $\{T\}^{\text{c-}value}_{\text{ideal}}$ and $\{T\}^{\text{c-}value}_{\text{Enlarged}}$, respectively. Similarly, the computed *MAP* over $\{T\}^{\text{YATEA}}_{\text{YATEA}}$ and $\{T\}^{\text{YATEA}}_{\text{Enlarged}}$ are, respectively, plotted in Figures 5.29 and 5.30. In these figures, the baselines are the average of the performances obtained when using the c-*value* and YATEA based rankings for extracting the aforementioned categories of terms. For example, if the set of annotated terms in the GENIA corpus that are ranked by the c-*value* measure (i.e., $\{T\}^{\text{c-}value}_{\text{ideal}}$), the computed NAP_i at 2% for the three categories of protein, cell line, and cell type are 0.37, 0.14, and 0.25, respectively. The mean average baseline is thus the sum of these numbers divided by three, which is ≈ 0.25 as reported in Figures 5.27a and 5.28a.

A series of discussions can follow the comparison of the plotted results in Figures 5.27, Figure 5.28, Figure 5.29, and Figure 5.30, similar to the approach employed in the previous sections. Evidently, depending on factors such as the targeted recall point and the characteristics of the corpus, the method's parameters can be tuned differently to obtain the best-averaged performances.

5.5 Discussion

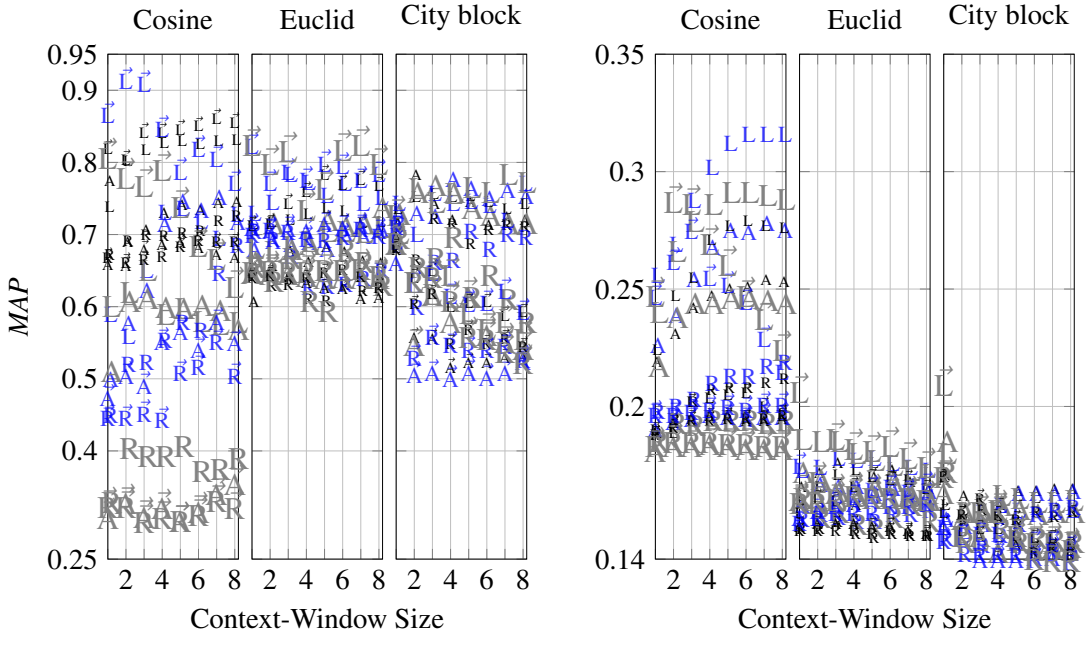
In Section 5.4, the use of the proposed distributional method for finding co-hyponym terms using a memory-based classification technique is investigated through a set of empirical experiments. Firstly, the results from these experiments allow one to accept the proposed hypothesis—that is, terms from a similar category of concepts appear in similar context, and that can be used for developing a distributional method for identifying co-hyponym terms. It is shown that with a small number of annotated reference terms (i.e., $|R_s| = 100$) and in the absence of sufficient training data for developing an entity tagger (i.e., as shown in Section 5.4.1.1), automatically constructed vector space models with reduced dimensionality can be used to address the proposed task with an acceptable performance (i.e., well above a general term recognition baseline, an entity tagger, and a random baseline). The result is satisfactory, particularly when the little amount of manual effort for developing a model is taken into consideration.

To address research questions proposed in Chapter 1 (Section 1.4), experiments are designed and carried out over the *Cartesian* product of a set of values for configuring the



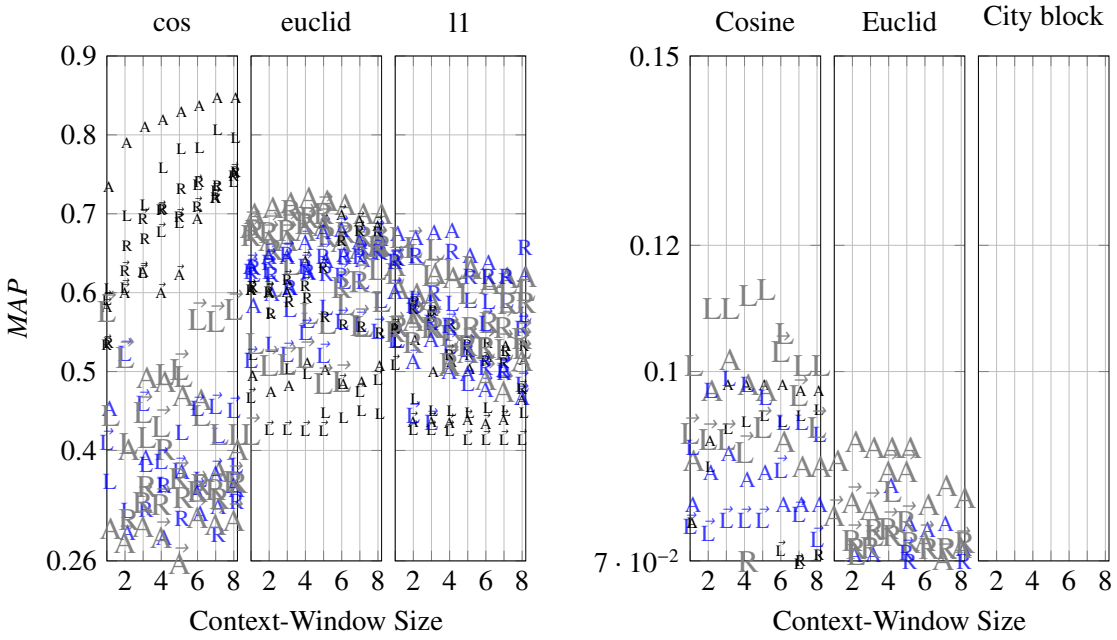
(a) $\boldsymbol{MAP_{2\%}}$ (b) $\boldsymbol{MAP_{100\%}}$

Figure 5.27: The mean average performance (i.e., y-axis) across concept categories observed in experiments over $\{\mathrm{T}\}^{\text{c-value}}_{\text{ideal}}$ using $|R_s| = 100$. The presentation format is similar to Figure 5.8: the letters show the direction in which the context-window is extended to collected co-occurrence frequency; their size (colour) denote the value of k; and the presence of $\vec{\square}$ on top of them indicates encoding information about the word order information.



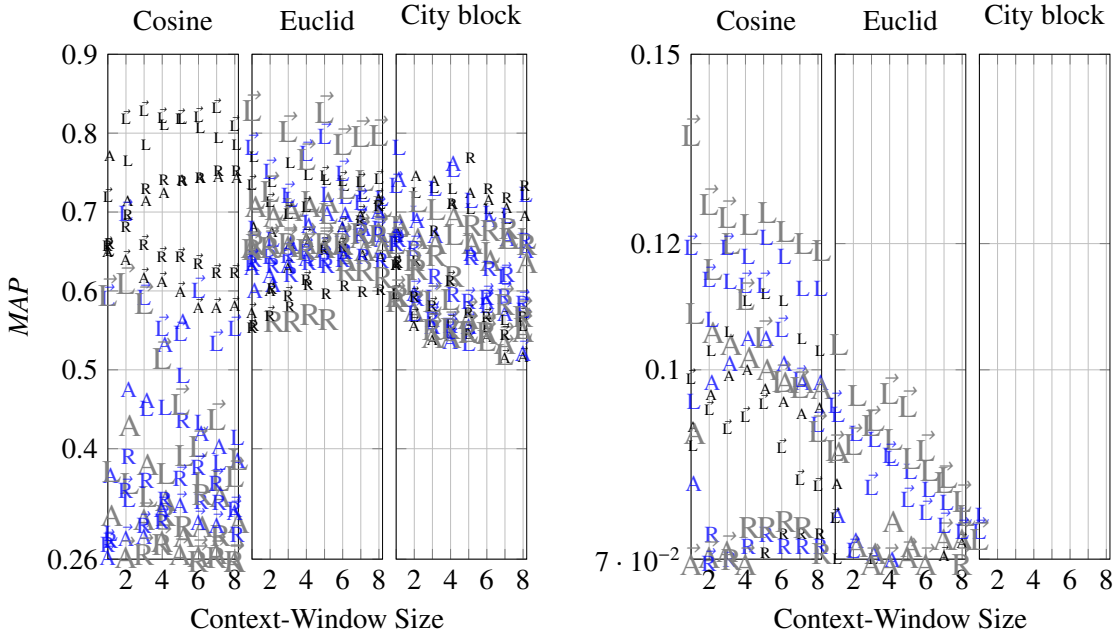
(a) $\boldsymbol{MAP_{2\%}}$ (b) $\boldsymbol{MAP_{100\%}}$

Figure 5.28: The mean average performance across concept categories observed in experiments over $\{\mathrm{T}\}^{\text{c-value}}_{\text{Enlarged}}$ using $|R_s| = 100$.



(a) $MAP_{2\%}$ (b) $MAP_{100\%}$

Figure 5.29: The mean average performance across concept categories observed in experiments over $\{\mathrm{T}\}^{\mathrm{YATEA}}_{\mathrm{YATEA}}$ using $|R_s| = 100$. At 100% recall, if the city block distance is employed to compute similarities, the method underperforms the computed baseline.



(a) $MAP_{2\%}$ (b) $MAP_{100\%}$

Figure 5.30: The mean average performance across concept categories observed in experiments over $\{\mathrm{T}\}^{\mathrm{YATEA}}_{\mathrm{Enlarged}}$ using $|R_s| = 100$. Similar to Figure 5.29, at 100% recall, the use of city block distance results in performances below the baseline.

parameters of the context-widow (i.e., to address RQ 1.1, 1.2, and 1.3) and the classification framework (i.e., RQ 2.1 and 2.2). To cover the remaining research questions, these experiments are repeated over several sets of candidate terms, and in corpora of two different sizes (i.e., to investigate RQ 3), in order to extract terms from various categories of concepts (i.e., in pursuing RQ 4). The non-interpolated average precisions at two recall points (2% and 100%) are reported as the figure of merit.

To address research questions about the configuration of context-windows, several models are constructed when the context-windows are extended to three different *directions*: only to the left, only to the right, and in both directions around the candidate terms (see RQ 1.1); with variable *sizes* of $1 \leq t \leq 8$ tokens (see RQ 1.2); and when the *sequential order* of words in the context-windows are encoded and neglected (see RQ 1.3). Hence, 48 different models are constructed for each set of candidate terms and each corpus employed in the experiments. To address questions about the parameters of the similarity-based reasoning framework, the weighting process is carried out using three *similarity metrics*: the city block distance, the cosine measure, and the Euclidean distance (see RQ 2.1). This is done for three different values of the *neighbourhood size* k (see RQ 2.2)—therefore, the categorisation process is repeated for $k \in 1, 7, 25$.

In Section 5.4.1, the experiments begin with the evaluation of the method for identifying terms from the category of *proteins* in the constructed terminological resource from the GENIA corpus and using this corpus for collecting the co-occurrence frequencies (i.e., $\{\mathrm{T}\}^{\textit{c-value}}_{\text{ideal}}$, which is free from invalid candidate terms). Accordingly, the method's performances are obtained when it is configured using the aforementioned values for its parameters. In Section 5.4.2, the experiments are repeated in the same corpus, however, using a set of candidate terms that are extracted using a state-of-the-art term extractor system (i.e., $\{\mathrm{T}\}^{\mathrm{YATEA}}_{\mathrm{YATEA}}$, which contains invalid candidate terms).[1] In experiments that are performed over both $\{\mathrm{T}\}^{\textit{c-value}}_{\text{ideal}}$ and $\{\mathrm{T}\}^{\mathrm{YATEA}}_{\mathrm{YATEA}}$, it is observed that choosing the best performing configuration is largely dependant on the recall value that is targeted.

While it is not possible to choose a one best value for the size of context-windows, it is verified that extending the context-windows to more than 5 tokens does not improve the computed performances, particularly for large recall values. With respect to the direction in which context-windows are extended to collect co-occurrences, the conclusion is similar: depending on the employed similarity metric and the targeted recall value, the best performing models are constructed when they are stretched in different directions. However, more than often, context-windows extended to the left of candidate terms outperform context-windows that are stretched in the other directions. However, in experiments over $\{\mathrm{T}\}^{\mathrm{YATEA}}_{\mathrm{YATEA}}$, specially when using the cosine measure, the context-windows that extend around the candidate terms can outperform those that extend only to the left. As discussed in Section 5.4.2, one explanation for this observation is that invalid terms in $\{\mathrm{T}\}^{\mathrm{YATEA}}_{\mathrm{YATEA}}$ often contain valid terms that appear nested at one side of invalid terms.

A similar conclusion can be drawn for deciding upon the inclusion of information about the order of words in context-windows. It is shown that word order information does not necessarily enhance the observed performances (see Figures 5.9b and 5.18). Distance metrics are understood to respond differently to the inclusion of word order

[1] See Table 5.4.

information (i.e., if this information improves the performance when using one of them, it does not necessarily enhance the result when using the other one). Apart from the employed metric for similarity measurement and the configuration of context-windows, the results show that the targeted recall value is an important factor when deciding on the inclusion of word order information. For small recall values, in many experiments, context-windows that encode information about the order of words are among the top performers.

The discussion about the parameters of the classification framework (see RQ 2.1 as well as RQ 2.2) is comparable to the discussion about the configuration of context-windows in the sense that the targeted recall value plays an important role in choosing the best performing configuration. In general, for small recall values—and, when the number of extracted terms is not much larger than $|R_s|$—the Euclidean and the city block distance metrics perform better than the cosine measure. However, the performance of similarity measures that are based on the distance metrics drops abruptly for large recall values. The cosine measure thus seems to be a preferable choice in the majority of applications. Particularly, the cosine measure seems to have a more stable behaviour in the sense that a higher correlation between the observed results is obtained when the set of candidate terms are altered (i.e., when the performances obtained in $\{\mathrm{T}\}^{\text{c-}value}_{\text{ideal}}$ and $\{\mathrm{T}\}^{\text{YATEA}}_{\text{YATEA}}$ are compared). Concerning the neighbourhood selection value (i.e., k), in the majority of the experiments and on average, a large value (i.e., $k = 25$) shows a better performance than a small value such as $k = 1$, or 7. However, when using the cosine measure and for small recall values, a small value of k can result in a higher performance than a large k.

In order to investigate the effect of the corpus size in the method's performance (see RQ 3), the experiments are continued by fetching additional text and enlarging the GENIA corpus from half a million to 55 million tokens. In turn, as reported in Section 5.4.3, the interplay between the size of the corpus that is used for the construction of the models, the configuration of context-windows (i.e., the way co-occurrence frequencies are collected), and the metrics that are employed to measure similarity between vectors is investigated.

The experiments show that increasing the size of the input corpus for collecting co-occurrence frequencies can improve the performance of the method if a suitable configuration of context-windows and classification parameters (particularly, similarity metric) are employed. It is observed that the top performer parameters in the original corpus of a small size are not necessarily the top performers when the corpus size increases. In addition, it is noticed that choosing the best performing parameters largely depends on the criteria set for the performance assessment. For instance, the city block distance showed a poor performance when the method is assessed at 100% recall. However, at a small recall point, the city block showed a superior performance. These observations can perhaps justify a number of contradictory reports in literature on the effect of the corpus size in the performance of distributional models.

On average, compared to the Euclidean and the city block distance, cosine showed a better performance and a more positive and stable response to an increases in the size of the input corpus. This result can be expected intuitively, since cosine shows the degree of commonality between the elements of two vectors. One can suspect that frequency normalisation and smoothing can enhance the results when using the Euclidean distance. How-

ever, an initial experiment to investigate this mater has resulted in even poorer results.[1] The entries of specialized vocabularies are rare and less frequent than general vocabularies. For example, a handful of terms in the GENIA corpus (e.g., the term *physiologic cell lineage*) are so rare that they have appeared only once in the enlarged corpus. Hence, enlarging the corpus will not change the collected co-occurrence frequencies for a relatively large number of terms (see Figure 5.13).

Lastly, to investigate the method's performance across categories of concepts (see RQ 4), the evaluation of the method is extended to a few categories of co-hyponym terms in the GENIA corpus. In Section 5.4.4, it is shown that despite similarities in the configurations of the method that give the best performances for identifying terms in each category (e.g., as shown in Figures 5.23, 5.24, 5.25, and 5.26, using context-windows that extend to the left of candidate terms and the cosine measure for computing similarities often results in the best observed performances), suggesting that it is not possible to recommend a one best configuration for context-windows and the classification parameters across all the categories.

This observation can be utilised when a clustering technique (e.g., as proposed in Dupuch et al., 2014) is employed for identifying co-hyponym terms. The aforementioned observation—that is, the performance of the method, particularly, with respect to the configuration of context-windows is different from one co-hyponym category to another—is often overlooked in these clustering tasks. That is to say, one single configuration of context-windows for collecting co-occurrence frequencies is employed to construct a single model and to perform the clustering process. Using several models in parallel that are constructed by collecting co-occurrences from context-windows of different configurations could, perhaps, enhance the performance of these techniques. If this is not feasible (e.g., due to the lack of computational resources), then a model can be chosen by averaging the performances across categories of concepts, such as the one proposed in Section 5.4.5.

5.6 Improving the Performance for Large Recall Values

Tuning the evaluated parameters of the proposed method enhances the observed performances, particularly for small recall values. However, with the settings employed for its evaluation in the previous sections (particularly, using $|R_s| = 100$), the method suffers from a low performance (precision) when a large recall value (e.g., 100%) is desirable. This problem can be solved by additional reference vectors (i.e., training samples) and enlarging the size of R_s, for example, as reported in our experiments in Zadeh and Handschuh (2014b).

In the suggested method, the use of the example-based learning technique allows the addition of training instances and enlarging the R_s during the life cycle of the proposed system. Hence, in some applications, R_s can be extended *manually*, for example, through iterative interactions between the user and the system. Whereas this can be a reasonable solution in a number of use cases, it still may not be favourable in some situations. In this

[1] The results from these experiments are thus not reported in details.

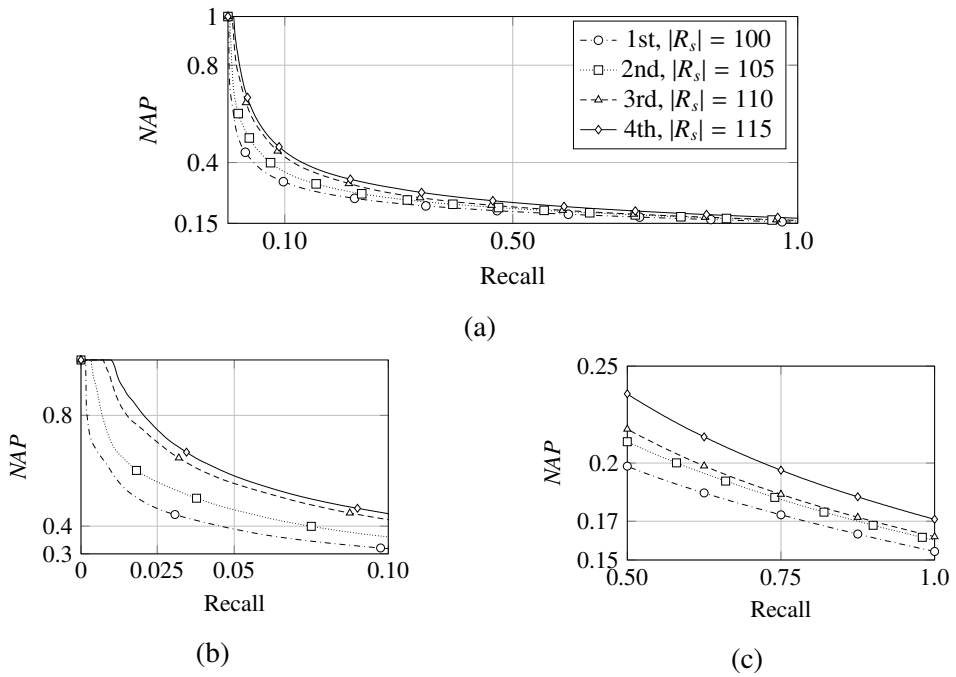


Figure 5.31: Bootstrap learning: the observed non-interpolated precision (*NAP*) for four iterations in the performed experiment: (a) plots the observed *NAP* (i.e., *y*-axis) over the complete range of recall values (i.e., *x*-axis); (b) and (c) provide minute details. In each iteration, *NAP* improves slightly.

case, an alternative solution is the use of the *bootstrap learning methodology*.

In the bootstrap learning technique, the available annotated data (i.e., R_s) is used to train a classifier, and to label some of the unlabelled data. The resulting labelled data is then employed to extend the available training dataset, to develop a new model, and to label additional unlabelled data. This process is often repeated several times until no improvements are observed. In the context of natural language processing, this methodology is often known as the Yarowsky algorithm (Yarowsky, 1995). Despite errors that are inevitable due to the automatic expansion of the training data, which may limit the performance of this methodology (e.g., as addressed in McIntosh and Curran, 2009),[1] the Yarowsky algorithm has been applied successfully to many information extraction problems.

In the proposed co-hyponym identification task, the Yarowsky algorithm can be employed to resolve the problem of low performance at large recall points. Originally, Yarowsky proposed his unsupervised learning algorithm for word sense disambiguation based on the observation that words often express only one major sense in a given discourse or document. As stated earlier, in special corpora, the proportion of polysemous terms is very low (e.g., 4% in the GENIA corpus vs. 17% in WordNet). Evidently, for the proposed co-hyponym term extraction task, the prerequisite condition for a successful

[1] Often known as the problem of *concept drifting*, or *semantic drifting*.

application of the Yarowsky algorithm is met.

Whereas the study of this algorithm is well beyond the scope of this thesis (e.g., see discussions in Abney, 2004, for an in-depth understanding of the important parameters in the Yarowsky algorithm), as a proof of concept, the observed results from a limited experiment, which is performed over the $\{\mathrm{T}\}^{\mathrm{Y_A T_E A}}_{\mathrm{Y_A T_E A}}$, are reported. In this experiment, for a particular configuration of context-windows (i.e., using context-windows of size three tokens that are extended to the left of candidate terms), and classification parameters (i.e., using the cosine similarity and the $k = 25$), the classification process is repeated for several iterations. In each iteration, after ranking the candidate terms by their assigned weight in that iteration, the top five candidate terms are added as positive examples to R_s. Figure 5.31 shows the observed results in the first four iterations. As shown, in each iteration, the performance of the method improves slightly.

5.7 Summary

In this chapter, the main method for identifying co-hyponym terms is proposed and evaluated. Terminological resources are often structured by organising terms into a number of categories in the domain of expertise that they represent. In Section 5.1, it is described that terms that are placed under each category of concepts are in a co-hyponymy relationship, which can be modelled—linguistically—as a kind of paradigmatic relationship. Consequently, it is explained that the principles of automatic term recognition (explained in Chapter 3) and distributional semantics (described in Chapter 2) can be combined to extract co-hyponym terms.

Section 5.2 details the method. After the extraction of candidate terms, they are represented in vector space models that are constructed automatically by collecting their co-occurrence frequencies with words appear in narrow context-windows in their vicinity. Exploiting this method, however, is hindered by the high dimensionality of vector spaces—that is, the curse of dimensionality problem (see also RQ 5). To tackle this problem, based on the principles introduced in Chapter 4, random projections are employed for the incremental construction of vectors spaces with a reduced dimensionality. In turn, in these vector space models, the task of identifying co-hyponym terms is accomplished by using an example-based k-nearest neighbours learning framework and a small number of annotated terms as reference vectors R_s, of which $|R_s| = 100$.

As discussed in Section 5.3.5, a number of factors play roles in the performance of the proposed method: (a) the configuration of context-windows for the collection of co-occurrence frequencies; and, (b) setting the parameters of the learning framework—that is, the neighbourhood size (k) selection and the employed metric for similarity measurements. These parameters are evaluated systematically in Section 5.4. Apart from the influence of these parameters on the performance of the method for identifying co-hyponym terms, the method's performance is studied with respect to (a) the presence of noise (i.e., invalid terms) in the list of candidate terms (see Section 5.4.2), and (b) the size of input corpus for collecting co-occurrence frequencies (i.e., enlarging the corpus as described in Section 5.4.3). In Section 5.4.4, the reported experiments are followed by investigating the method's performance across concept categories. In Section 5.5, the

results observed in these experiments are discussed and linked to the research questions proposed in Chapter 1.

Lastly, to improve the performance of the method when the extraction of co-hyponym terms at large recall values is desirable, Section 5.6 suggests the use of a bootstrap learning technique. It is proposed that an unsupervised learning method such as the Yarowsky algorithm can be employed to enlarge R_s iteratively, and thus enhance the observed performances across the complete range of recall values. To support the claim, the results observed in a limited number of experiments are reported.

This page is intentionally left blank.

Reference List

Abney, S. (2004). Understanding the Yarowsky algorithm. *Computational Linguistics*, 30(3):365–395. 190

Agirre, E., Alfonseca, E., Hall, K., Kravalova, J., Paşca, M., and Soroa, A. (2009). A study on similarity and relatedness using distributional and WordNet-based approaches. In *NAACL HLT 2009: Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 19–27, Boulder, Colorado. Association for Computational Linguistics. 152

Aubin, S. and Hamon, T. (2006). Improving term extraction with terminological resources. In Salakoski, T., Ginter, F., Pyysalo, S., and Pahikkala, T., editors, *Advances in Natural Language Processing*, volume 4139 of *Lecture Notes in Computer Science*, pages 380–387. Springer Berlin Heidelberg. 149

Balog, K. and de Rijke, M. (2008). Associating people and documents. In Macdonald, C., Ounis, I., Plachouras, V., Ruthven, I., and White, R. W., editors, *Advances in Information Retrieval*, volume 4956 of *Lecture Notes in Computer Science*, pages 296–308. Springer Berlin Heidelberg. 142

Baroni, M., Dinu, G., and Kruszewski, G. (2014). Don't count, predict! A systematic comparison of context-counting vs. context-predicting semantic vectors. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 238–247, Baltimore, US. Association for Computational Linguistics. 152

Buitelaar, P., Cimiano, P., and Magnini, B. (2005). Ontology learning from text: An overview. In Buitelaar, P., Cimiano, P., and Magnini, B., editors, *Ontology Learning from Text: Methods, Evaluation and Applications*, volume 123 of *Frontiers in Artificial Intelligence and Applications*, pages 3–12. IOS Press. 140

Buitelaar, P. and Eigner, T. (2008). Topic extraction from scientific literature for competency management. In Mochol, M., Zhdanova, A. V., Nixon, L., Breslin, J., and Polleres, A., editors, *Personal Identification and Collaborations: Knowledge Mediation and Extraction (PICKME 2008)*, volume 403, pages 55–66, Karlsruhe, Germany. CEUR Workshop Proceedings. 142

Bullinaria, J. A. and Levy, J. P. (2007). Extracting semantic representations from word co-occurrence statistics: A computational study. *Behavior Research Methods*, 39(3):510–526. 167

Chakraborty, S., Subramanian, L., and Nyarko, Y. (2014). Extraction of (key,value) pairs from unstructured ads. In *AAAI Fall Symposium Serie*, pages 10–17, Arlington, Virginia. AAAI Press. 142

Cimiano, P., Hotho, A., and Staab, S. (2005). Learning concept hierarchies from text corpora using formal concept analysis. *Journal of Artificial Intelligence Research*, 24:305–339. 140

Cimiano, P., McCrae, J., Buitelaar, P., and Montiel-Ponsoda, E. (2013). On the role of senses in the ontology-lexicon. In Oltramari, A., Vossen, P., Qin, L., and Hovy, E., editors, *New Trends of Research in Ontologies and Lexical Resources*, Theory and Applications of Natural Language Processing, pages 43–62. Springer Berlin Heidelberg. 141, 142

Daelemans, W. and van den Bosch, A. (2010). Memory-based learning. In Clark, A., Fox, C., and Lappin, S., editors, *The Handbook of Computational Linguistics and Natural Language Processing*, pages 154–179. Wiley-Blackwell. 144

Dupuch, M., Dupuch, L., Hamon, T., and Grabar, N. (2014). Exploitation of semantic methods to cluster pharmacovigilance terms. *Journal of Biomedical Semantics*, 5(18). 140, 188

Etzioni, O., Cafarella, M., Downey, D., Popescu, A.-M., Shaked, T., Soderland, S., Weld, D. S., and Yates, A. (2005). Unsupervised named-entity extraction from the Web: An experimental study. *Artificial Intelligence*, 165(1):91–134. 141

Frantzi, K. T., Ananiadou, S., and Tsujii, J. (2000). The C-value/NC-value method of automatic recognition for multi-word terms. In *Research and Advanced Technology for Digital Libraries*, volume 3 of *Lecture Notes in Computer Science*, pages 585–604. Springer Berlin Heidelberg. 149

Gorman, J. and Curran, J. R. (2006). Scaling distributional similarity to large corpora. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL*, pages 361–368, Sydney. Association for Computational Linguistics. 167

Harris, Z. S. (1954). Distributional structure. *Word, The Journal of the International Linguistic Association*, 10:146–162. 143

Hastie, T., Tibshirani, R., and Friedman, J. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer Series in Statistics. Springer Science+Business Media, 2nd edition. 154

Hearst, M. A. (1992). Automatic acquisition of hyponyms from large text corpora. In Boitet, C., editor, *Proceedings of the fifteenth International Conference on Computational Linguistics: COLING-92*, volume II, pages 539–545, Nantes, France. GETA (IMAG) and Association Champollion / Association for Computational Linguistics. 140, 143

Jones, M. N. and Mewhort, D. J. K. (2007). Representing word meaning and order information in a composite holographic lexicon. *Psychological Review*, 114(1):1–37. 152

Kim, J.-D., Ohta, T., Tateisi, Y., and Tsujii, J. (2003). GENIA corpus—a semantically annotated corpus for bio-textmining. *Bioinformatics*, 19(Suppl 1):i180–i182. 142, 148

Kim, J.-D., Ohta, T., Tateisi, Y., and Tsujii, J. (2006). GENIA corpus manual. Technical Report TR-NLP-UT-2006-1, Tsujii Laboratory. 148

Kim, J.-D., Ohta, T., Tsuruoka, Y., Tateisi, Y., and Collier, N. (2004). Introduction to the bio-entity recognition task at JNLPBA. In Collier, N., Ruch, P., and Nazarenko, A., editors, *JNLPBA: Proceedings of the International Joint Workshop on Natural Language Processing in Biomedicine and Its Applications*, pages 70–75, Geneva, Switzerland. Association for Computational Linguistics. 140, 148, 161

Landauer, T. K. (2002). On the computational basis of learning and cognition: Arguments from LSA. *The Psychology of Learning and Motivation*, 41:43–84. 152, 153

Lenci, A. (2008). Distributional semantics in linguistic and cognitive research. *From Context to Meaning: Distributional Models of the Lexicon in Linguistics and Cognitive Science, Special Issue of the Italian Journal of Linguistics*, 20(1):1–31. 152

L'Homme, M.-C. and Bernier-Colborne, G. (2012). Terms as labels for concepts, terms as lexical units: A comparative analysis in ontologies and specialized dictionaries. *Applied Ontology*, 7(4):387–400. 140

Manning, C. D., Raghavan, P., and Schütze, H. (2009). *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA, draft april 1, 2009 edition. 183

Maynard, D., Funk, A., and Peters, W. (2009). Using lexico-syntactic ontology design patterns for ontology creation and population. In Blomqvist, E., Sandkuhl, K., Scharffe, F., and Svatek, V., editors, *Workshop on Ontology Patterns: WOP 2009: Papers and Patterns from the ISWC workshop*, volume 516, pages 39–52, Washington D.C., USA. CEUR Workshop Proceedings. 140

McIntosh, T. and Curran, J. R. (2009). Reducing semantic drift with bagging and distributional similarity. In *ACL-IJCNLP 2009: Joint Conference of the 47th Annual Meeting*

of the Association for Computational Linguistics and 4th International Joint Conference on Natural Language Processing of the AFNLP: Proceedings of the Conference, volume 1, pages 396–404, Suntec, Singapore. Association for Computational Linguistics and the Asian Federation of Natural Language Processing. 189

Mihalcea, R. (1998). Semcor. Available for download from http://web.eecs.umich.edu/~mihalcea/downloads/semcor/semcor3.0.tar.gz. 142

Miller, G. A. (1995). WordNet: A lexical database for English. *Communications of the ACM*, 38(11):39–41. 142

Pantel, P., Crestan, E., Borkovsky, A., Popescu, A.-M., and Vyas, V. (2009). Web-scale distributional similarity and entity set expansion. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 938–947, Singapore. Association for Computational Linguistics. 167

QasemiZadeh, B. (2010). Towards technology structure mining from text by linguistics analysis. Tech Report DERI TR 010-02-15, Digital Enterprise Research Institute, Lower Dangan, Galway. 143

QasemiZadeh, B. (2015). *Investigating the Use of Distributional Semantic Models for Co-Hyponym Identification in Special Corpora*. PhD thesis, National University of Ireland, Galway. i

Recchia, G., Sahlgren, M., Kanerva, P., and Jones, M. N. (2015). Encoding sequential information in semantic space models: Comparing holographic reduced representation and random permutation. *Computational Intelligence and Neuroscience*, 2015:1–18. 152

Rijsbergen, C. J. V. (1977). A theoretical basis for the use of co-cccurrence data in information retrieval. *Journal of Documentation*, 33(2):106–119. 142

Sahlgren, M. (2006). *The Word-Space Model: Using distributional analysis to represent syntagmatic and paradigmatic relations between words in high-dimensional vector spaces*. PhD thesis, Stockholm University. 143

Sahlgren, M., Holst, A., and Kanerva, P. (2008). Permutations as a means to encode order in word space. In Sloutsky, V., Love, B., and Mcrae, K., editors, *Proceedings of the 30th Annual Conference of the Cognitive Science Society*, pages 1300–1305. Cognitive Science Society, Austin, TX. 152, 154

Schütze, H. (1993). Word space. In Hanson, S., Cowan, J., and Giles, C., editors, *Advances in Neural Information Processing Systems 5 (NIPS 1992)*, pages 895–902, San Francisco, CA, USA. Morgan-Kaufmann. 143

Settles, B. (2005). ABNER: An open source tool for automatically tagging genes, proteins and other entity names in text. *Bioinformatics*, 21(14):3191–3192. 161

Weeds, J., Dowdall, J., Schneider, G., Keller, B., and Weir, D. (2005). Using distributional similarity to organise BioMedical terminology. *Terminology*, 11(1):3–4. 155

Wong, W., Liu, W., and Bennamoun, M. (2012). Ontology learning from text: A look back and into the future. *ACM Computing Surveys*, 44(4):20:1–20:36. 140

Yarowsky, D. (1995). Unsupervised word sense disambiguation rivaling supervised methods. In *33rd Annual Meeting of the Association for Computational Linguistics: Proceedings of the Conference*, pages 189–196, Cambridge, Massachusetts, USA. Association for Computational Linguistics. 189, 190, 191

Yi, K. (2010). A semantic similarity approach to predicting library of congress subject headings for social tags. *Journal of the American Society for Information Science and Technology*, 61(8):1658–1672. 142

Zadeh, B. Q. and Handschuh, S. (2014a). Evaluation of technology term recognition with random indexing. In Calzolari, N., Choukri, K., Declerck, T., Loftsson, H., Maegaard, B., Mariani, J., Moreno, A., Odijk, J., and Piperidis, S., editors, *Proceedings of the Ninth International Conference on Language Resources and Evaluation*, Reykjavik, Iceland. European Language Resources Association. ACL Anthology Identifier: L14-1703. 139

Zadeh, B. Q. and Handschuh, S. (2014b). Investigating context parameters in technology term recognition. In Meyers, A., He, Y., and Grishman, R., editors, *Proceedings of the COLING Workshop on Synchronic and Diachronic Approaches to Analyzing Technical Language (SADAATL 2014)*, pages 1–10. Association for Computational Linguistics and Dublin City University. 139, 152, 188

This page is intentionally left blank.