A Minwise Hashing Method for Addressing Relationship Extraction from Text

David S. Batista, Rui Silva, Bruno Martins, and Mário J. Silva

Instituto Superior Técnico and INESC-ID, Lisboa, Portugal
{david.batista,rui.teixeira.silva,bruno.g.martins,mario.gaspar.silva}@ist.utl.pt

Abstract. Relationship extraction concerns with the detection and classification of semantic relationships between entities mentioned in a collection of textual documents. This paper proposes a simple and on-line approach for addressing the automated extraction of semantic relations, based on the idea of nearest neighbor classification, and leveraging a minwise hashing method for measuring similarity between relationship instances. Experiments with three different datasets that are commonly used for benchmarking relationship extraction methods show promising results, both in terms of classification performance and scalability.

Keywords: Text Mining, Relationship Extraction, Minwise Hashing

1 Introduction

The task of relationship extraction concerns with the detection and classification of semantic relationships between entities mentioned in a collection of textual documents. Popular application domains include the detection of gene-disease relationships or protein-protein interactions in biomedical literature [4,11,19], the detection of associations between named entities referenced in news or web corpora (e.g., birthplace relations between persons and locations, or affiliation relations between persons and organizations) [3,6], or the detection of relations between pairs of nominals in general [9].

Over the years, multiple approaches have been proposed to address relationship extraction [9,11,17]. Rule-based methods employ a number of linguistic rules to capture relation patterns. Feature-based methods, on the other hand, transform the text into a large amount of linguistic features (e.g., lexical, syntactic and semantic features), later capturing the similarity between these feature vectors through traditional supervised learning methods. Recent developments have mainly relied on kernel-based learning approaches, either exploring kernels for representing sequences [4], in an attempt to capture sequential patterns within sentences, or kernels specific for trees or graph structures in general, to learn features related to parse tree structures [3,13]. Kernel methods are better than feature-based methods at circumventing data sparseness issues and at exploring very large feature spaces, but nonetheless they are also computationally demanding. Whenever one needs to address real-world problems involving hundreds of

relationship classes as expressed on large amounts of textual data, and when dealing with large training datasets, scalability becomes an issue.

In this paper we explore the automated extraction of semantic relations, based on nearest neighbor classification. To make the nearest neighbor search computationally feasible, we leverage an efficient method based on minwise hashing and on Locality-Sensitive Hashing (LSH) [2,15]. Experiments with three different collections that are commonly used for benchmarking relationship extraction methods, namely the dataset from the SemEval 2010 task on multi-way classification of semantic relations between pairs of nominals [9], the Wikipedia relation extraction dataset created by Culotta et al. [6], and the AImed dataset of human protein interactions [4], showed good results. We specifically tested different configurations of the proposed method, varying the minwise hashing signatures and the number of considered nearest neighbors. Our best results correspond to a macro-averaged F1 score of 0.69 on the SemEval dataset, a macro-averaged F1 score of 0.43 on Wikipedia, and an F1 score of 0.52 on AImed. These values come close to the state-of-the-art results reported for these datasets, and we argue that the method has advantages in simplicity and scalability.

Section 2 of the paper presents related work. Section 3 details the proposed method, describing the considered representation for the relation instances, and presenting the minwise hashing approach that was used. Section 4 presents the experimental evaluation of the proposed method. Finally, Section 5 summarizes our conclusions, and outlines possible directions for future work.

2 Related Work

Extracting semantic relations between nominal expressions (e.g., named entities like persons, locations or organizations) in natural language text is a crucial step towards document understanding, with many practical applications. Several authors have addressed the problem, for instance by formulating it as a binary classification task (i.e., classifying candidate instances of binary relations, between pairs of nominal expressions, as either related or not). Relevant previous approaches include those that adopt feature-based supervised learning methods [10,21], or kernel-based methods [17,18] to perform relation extraction. The major advantage of kernel methods is that they allow one to explore a large (often exponential or, in some cases, infinite) feature space in polynomial computational time, without the need to explicitly represent the features. Nonetheless, kernel methods are still highly demanding in terms of computational requirements, whenever one needs to manipulate large training data sets. The main reason for this lays in the fact that kernel methods, even if relying only on very simple kernels, are typically used together with models and learning algorithms such as Support Vector Machines (SVM), where training involves a quadratic programming optimization problem and is typically performed off-line. Moreover, given that SVMs can only directly address binary classification problems, it is necessary to train several classifiers (i.e., in a one-versus-one or a one-versus-all strategy) to address multi-class relation extraction tasks.

Given a set of positive and negative binary relation examples, feature-based methods start by extracting syntactic and semantic features from the text, using them as cues for deciding whether the entities in a sentence are related or not. Syntactic features extracted from the sentences include (i) the entities themselves, (ii) the semantic types of the entities, (iii) the word sequence between the entities, (iv) the number of words between the entities, and (v) the path in a parse-tree containing the two entities. Semantic features can for instance include the path between the two entities in a dependency tree. The features are presented to a classifier in the form of a feature vector, which then decides on the relation class. Previous works have explored different types of supervised learning algorithms and different feature sets [10,21].

Feature-based methods have the limitation of involving heuristic choices, with features being selected on a trial-and-error basis, to maximize performance. To remedy the problem of selecting a suitable set of features, specialized kernels have been designed for relationship extraction. They leverage rich representations of the input data, exploring the input representations exhaustively, in an implicit manner and conceptually in a higher dimensional space.

For instance, Bunescu and Mooney presented a generalized subsequence kernel that works with sparse sequences, containing combinations of words and parts-of-speech (POS) tags to capture the word-context around the nominal expressions [4]. Three subsequence kernels are used to compute the similarity between sequences (i.e., between relation instances) at the word level, namely comparing sequences of words occurring (i) before and between, (ii) in the middle, and (iii) between and after the nominal expressions. A combined kernel is simply the sum of all three sub-kernels. The authors evaluated their approach on the task of extracting protein interactions from MEDLINE abstracts contained in the AImed corpus, concluding that subsequence kernels in conjunction with SVMs improve both precision and recall, when compared to a rule based system. Bunescu and Mooney also argued that, with this approach, augmenting the word sequences with POS tags and entity types can lead to better results than those obtained with the dependency tree kernel by Culotta and Sorensen [7].

Zelenko et al. described a relation extraction approach, based on SVMs or Voted Perceptrons, which uses a tree kernel defined over a shallow parse tree representation of the sentences [18]. The kernel is designed to compute the similarity between two entity-augmented shallow parse tree structures, in terms of a weighted sum of the number of subtrees that are common between the two shallow parse trees. These authors evaluated their approach on the task of extracting *person-affiliation* and *organization-location* relations from text, noticing that the proposed method is vulnerable to unrecoverable parsing errors.

Culotta and Sorensen described a slightly generalized version of the previous kernel, based on dependency trees, in which a bag-of-words kernel is also used to compensate for errors in syntactic analysis [7]. Every node of the dependency tree contains rich information like word identity, POS and generalized-POS tags, phrase types (noun-phrase, verb-phrase, etc.), or entity types. Using a rich structured representation can lead to performance gains, when compared to bag-of-

words approaches. A further extension is proposed by Zhao and Grishman, using composite kernels to integrate information from different syntactic sources [20]. They incorporate tokenization, parsing, and dependency analysis, so that processing errors occurring at one level may be overcome by information from other levels. Airola et al. introduced the all-dependency-paths kernel [1]. They use a representation based on a weighted directed graph that consists of two unconnected subgraphs, one representing the dependency structure of the sentence, and the other representing the sequential ordering of the words. Bunescu and Mooney presented yet another alternative approach which uses information concentrated in the shortest path, at a dependency tree, between the two entities [3]. These authors argue that the shortest path between the two nominals encodes sufficient information to infer the semantic relation between them.

Recent studies continue to explore combinations or extensions of the previously described kernel methods [11,13]. However, most proposals have been evaluated on different data sets, making it difficult to assess which is better.

3 Relationship Extraction as Similarity Search

The proposed approach for classifying pairs of substrings (e.g., pairs of nominal expressions) from a given sentence, according to the semantic relation that the sentence expresses over these substrings, is based on the idea of finding the most similar relation instances from a given database of examples. A representation for each candidate binary relation, obtained from the sentence where the pairs of substrings co-occur, is first generated. We then assign the relationship type, to the instance being classified, according to the most frequent/similar type at the top k most similar relation instances, gathered from the database of examples. This procedure essentially corresponds to a weighted kNN classifier, where each example instance has a weight corresponding to its similarity towards the instance being classified, and where the more similar instances have therefore a higher vote, in the classification, than the ones that are more dissimilar.

The considered representation for each binary relation instance is essentially based on character quadgrams, specifically considering (i) the character quadgrams occurring between the two operands that constitute the binary relation (i.e., between the two substrings corresponding to the nominals that are related), (ii) the quadgrams occurring in a window of three tokens before the first operand and between operands, and (iii) the quadgrams occurring between operands and in a window of three tokens after the second operand. Consider, for instance, the following sentence, where the related nominals are in bold: *the* ***micropump*** *is fabricated by* ***anisotropic etching****, considering orientation.* The substring (i) would correspond to *is fabricated by the*, while substring (ii) would correspond to *the micropump is fabricated by*, and substring (iii) would correspond to *is fabricated by anisotropic etching, considering orientation.*

This representation follows the observation that a relation between two entities is generally expressed using only words that appear in one of three basic patterns, namely before-between (i.e., tokens before and between the two entities

Dataset	Relationship	Relational Patterns
SemEval	cause-effect	caused by; have; caused; result in; triggered by;
	entity-origin	have; come from; be from; run away from; arrive from;
Wikipedia	member-of	play for; have also; serve in; elect to; serve as;
	award	nominate for; award; win; receive; run;
AImed	related	bind to; bind; have; interact with; do
	not-related	bind to; bind; have; do; may

Table 1: Frequent relational patterns associated to different relation types.

involved in the relation), between (i.e., only tokens between the two entities), and between-after (i.e., tokens between and after the two entities) [4]. Besides character quadgrams, we also use prepositions, verb forms in the past participle tense, the infinitive forms of verbs, and a relational pattern corresponding to a verb, followed by nouns, adjectives, or adverbs, and ending with a preposition. The previous relational pattern is inspired by one of the features used in ReVerb, an unsupervised open-domain relation extraction system [8]. The morphological information is extracted from the three same textual windows considered for the quadgrams (i.e., the substrings before-between, between, and between-after), with the help of the MorphAdorner NLP package[1].

We experimented with other representations for the relation instances, using different textual windows and different n-gram sizes, as well as with n-grams of tokens, after normalizing the text through lowercasing, lemmatization and/or WordNet-based generalization operations. However, the features described before achieve the best trade-off between accuracy and computational performance.

Each quadgram/token, at each of the three groups (i.e., before-between, between, and between-after), is assigned to a globally unique identifier. The similarity between two instances is measured through the Jaccard similarity coefficient between each set of globally unique identifiers.

For illustration purposes, Table 1 shows the top 5 most frequent relational patterns for two different relations in each of the three datasets that were used in our experiments – see Section 4. It is important to notice that in the case of the AImed dataset, similar relational patterns are extracted for both classes. We noticed that in instances from the *not-related* class, it is often the case that patterns such as *bind to* are preceded by different kinds of negation patterns.

The naive approach to finding the most similar pairs of relation instances, in a database of size N, involves computing all N^2 pairwise similarities, which quickly becomes a bottleneck for large N. Even if the task is parallelizable, overcoming this complexity is necessary to achieve good scalability, and it is therefore highly important to devise appropriate pre-processing operations that facilitate relatedness computations on the fly. In our case, this is done by approximating the Jaccard similarity coefficient through a minwise hashing (i.e., minhash) procedure, later leveraging a Locality-Sensitive Hashing (LSH) method for rapidly finding the kNN most similar relation instances. We therefore have

[1] `http://morphadorner.northwestern.edu/`

that, contrary to traditional kNN classifiers, where training takes virtually zero computation time (i.e., it just involves storing the example instances) but classification is highly demanding, our approach involves a LSH method for indexing the training instances, which allows classification to be made efficiently, since we only have to measure similarity towards a small set of candidates. Similarly to other kNN classifiers, our approach remains relatively simple, performs the learning in an on-line fashion, and can directly address multi-class problems.

The min-hash technique was first introduced in the seminal work of Broder, where it was successfully applied to duplicate Web page detection [2]. Given a vocabulary Ω of size D (the set of all representative elements occurring in a collection of relation instances) and two sets, S_1 and S_2, where $S_1, S_2 \subseteq \Omega = \{1, 2, \ldots, D\}$, we have that the Jaccard similarity coefficient, between the sets of elements, is given by the ratio of the size of the intersection between S_1 and S_2, to the size of the union of both datasets:

$$\mathrm{J}(S_1, S_2) = \frac{|S_1 \cap S_2|}{|S_1 \cup S_2|} = \frac{|S_1 \cap S_2|}{|S_1| + |S_2| - |S1 \cap S_2|} \tag{1}$$

The two sets are more similar when their Jaccard index is closer to one, and more dissimilar when their Jaccard index is closer to zero. In large datasets, efficiently computing the set sizes is challenging, given that the total number of possible elements is huge. However, suppose a random permutation π is performed on the ordering that is considered for the elements in the vocabulary Ω, i.e., suppose $\pi : \Omega \longrightarrow \Omega$. An elementary probability argument shows that the Jaccard coefficient can be estimated from the probability of the first (i.e., the minimum) values of the random permutation π, for sets S_1 and S_2, being equal, given that the Jaccard coefficient is the number of common elements to both sets over the number of elements that exist in at least one of the sets.

$$\Pr\left(\min(\pi(S_1)) = \min(\pi(S_2))\right) = \frac{|S_1 \cap S_2|}{|S_1 \cup S_2|} = \mathrm{J}(S_1, S_2) \tag{2}$$

After the creation of k minwise independent permutations (i.e., $\pi_1, \pi_2, \ldots, \pi_k$) one can efficiently estimate $\mathrm{J}(S_1, S_2)$, without bias, as a binomial distribution:

$$\hat{\mathrm{J}}(S_1, S_2) = \frac{1}{k} \sum_{j=1}^{k} 1(\min(\pi_k(S_1)) = \min(\pi_k(S_2))) \tag{3}$$

$$\mathrm{Variance}(\hat{\mathrm{J}}(S_1, S_2)) = \frac{1}{k} \hat{\mathrm{J}}(S_1, S_2) \left(1 - \hat{\mathrm{J}}(S_1, S_2)\right) \tag{4}$$

Equations 3 and 4 show, respectively, the expected value of the binomial distribution used for estimating the Jaccard coefficient from the k random permutations, and the corresponding variance, which decreases for larger values of k. The function one–if–true() in Equation 3 returns one if the two sets share the same minimum value, and zero otherwise.

In the implementation of the minwise hashing scheme, each of the independent permutations is a hashed value, in our case taking 32 bits of storage. Each

of the k independent permutations is associated to a polynomial hash function $h^k(x)$ that maps the members of Ω to distinct values. For any set S we take the k values of $h^k_{min}(S)$, i.e. the member of S with the minimum value of $h^k(x)$. The set of k values is referred to as the min-hash signature of an instance.

Efficient nearest neighbor search is implemented through a Locality-Sensitive Hashing (LSH) technique, that leverages the min-hash signatures to compress the relation instance representations into small signatures (i.e., to generate small signatures from the set of all character quadgrams, prepositions, verb forms in the past participle tense, infinitive forms of verbs, and relational patterns occurring before-between, between, and between-after the relation operands), at the same time preserving the expected similarity of any pair of instances. This technique uses L different hash tables (i.e., L different MapDB[2] persistent storage units), each corresponding to an n-tuple from the min-hash signatures, that we here refer to as a band. At classification time, we compute a min-hash signature for a given target instance and then consider any pair that hashed to the same bucket, for any of the min-hash bands, to be a candidate pair. We check only the candidate pairs for similarity, using the complete min-hash signatures to approximate the Jaccard similarity coefficient. This way, we avoid the pair-wise similarity comparisons against all example instances, thus performing the kNN classification in a efficient manner. Chapter 3 of the book by Rajaraman and Ullman details the use of minwise hashing with LSH techniques, in applications related to finding similar items [15].

A complete outline for the proposed method is therefore as follows: First, sets of character quadgrams, prepositions, verb forms in the past participle tense, infinitive forms of verbs, and relational patterns, are extracted from the substrings that occur before-between, between, and between-after the relation operands, for each possible relation at each sentence from a given database of examples. Then, a list of min-hashes are extracted from the sets generated in the first step. The min-hashes are split into bands, and hashed into the L different hash tables. At classification, when checking if a given binary relation, as expressed in some sentence, is being described, we start by also extracting the same set of features, from the substrings that occur before-between, between, and between-after two named entities to be considered the relation operands. A min-hash signature is then generated from this set of features. Relation instances, from the collection of examples, with at least one identical LSH band, are considered as candidates, and their Jaccard similarity coefficient towards the target instance is then estimated, from the available min-hashes. The candidates are sorted according to their similarity towards the target instance, and the most relevant relationship type, computed with basis on the weighted votes from the top kNN most similar instances, is returned as the identified semantic relationship type.

[2] `http://www.mapdb.org/`

	SemEval		Wikipedia		AImed
	Train	Test	Train	Test	Data
# Sentences	8,000	2,717	2,199	926	2,202
# Terms	137,593	46,873	49,721	20,656	75,878
# Relation classes	19	19	47	47	2
# Relation instances (except *not-related/other*)	6,590	2,263	15,963	6,386	1,000
# Nominals	16,001	5,434	5,468	2,258	4,084
Avg. sentence length (terms)	119.8	119.4	177.2	172.8	184.2
StDev. sentence length (terms)	45.0	44.4	104.5	100.1	98
Avg. instances/class	421	143	295.6	135.9	1,961.5
StDev. instances/class	317.5	105.5	1707.3	728.2	1,372.5
Max. instances/class (except *not-related/other*)	844	22	268	113	1,000
Min. instances/class	1	1	1	1	1,000

Table 2: Statistical characterization of the considered datasets.

4 Experimental Validation

The minwise hashing method proposed for performing semantic relation extraction was evaluated with three different document collections, from different domains, which are commonly used as benchmarks for this problem.

A statistical characterization of the three datasets used in our experiments is given in Table 2. The SemEval dataset[3] consists of 10,717 sentences, annotated according to 19 possible classes between two nominals in each sentence (9 non-symmetric relations types, such as *cause-effect* or *instrument-agency*, plus another label for denoting that no relationship is being expressed). The dataset is relatively balanced between the classes, and is split into training and testing subsets, with 8,000 instances for training and 2,717 for testing.

The Wikipedia dataset[4] consists of paragraphs from 441 Wikipedia pages, containing annotations for 4,681 relation mentions of 53 different relation types like *job-title*, *birth-place*, or *political-affiliation*. The dataset comes split into training and testing subsets, with about 70% of the paragraphs for testing, and the remaining 30% for training. In the Wikipedia dataset, the distribution of the examples per class is highly skewed: *job-title* is the most frequent relation (379 instances), whereas *grandmother* and *discovered* have just one example in the dataset. Moreover, although the full dataset contains annotations to the 53 different semantic relationship types, only 46 types are included in both the training and testing subsets, and still, of these 46 relation types, 14 of them have less than 10 examples. We therefore measured our results over a subset of 46 relationship types. In the experiments with this Wikipedia dataset, we only considered the problem of predicting the relationship type (i.e., classifying according to one of the 46 semantic relationship types, or as *other*), and not according to the direction of the relation. Of all the three datasets, this was the one that least fitted our general approach for modeling the relationship extraction task, and significant adaptations had to be made.

[3] `http://semeval2.fbk.eu/semeval2.php?location=tasks&taskid=11`
[4] `http://cs.neiu.edu/~culotta/data/wikipedia.html`

Dataset	Min Hash	1 kNN P	1 kNN R	1 kNN F1	3 kNN P	3 kNN R	3 kNN F1	5 kNN P	5 kNN R	5 kNN F1	7 kNN P	7 kNN R	7 kNN F1
SemEval (18 classes)	200/25	0.662	0.622	0.641	0.683	0.642	0.662	0.698	0.652	0.674	0.698	0.637	0.666
	200/50	0.662	0.621	0.640	0.683	0.643	0.662	0.698	0.651	0.673	0.698	0.636	0.666
	400/25	0.664	0.636	0.650	0.685	0.668	0.676	0.708	0.672	0.690	0.691	0.667	0.679
	400/50	0.663	0.635	0.649	0.684	0.664	0.674	**0.708**	0.674	**0.690**	0.694	0.670	0.682
	600/25	0.657	0.631	0.644	0.677	0.660	0.669	0.697	0.674	0.685	0.695	0.660	0.677
	600/50	0.657	0.631	0.644	0.676	0.658	0.667	0.699	**0.678**	0.688	0.694	0.664	0.678
	800/25	0.654	0.630	0.642	0.675	0.656	0.665	0.694	0.662	0.678	0.696	0.658	0.677
	800/50	0.654	0.632	0.643	0.677	0.658	0.667	0.698	0.665	0.681	0.696	0.658	0.676
Wikipedia (46 classes)	200/25	0.410	0.336	0.369	0.434	0.335	0.378	0.439	0.310	0.363	0.489	0.323	0.389
	200/50	0.409	0.336	0.369	0.435	0.336	0.379	0.440	0.310	0.364	0.489	0.321	0.387
	400/25	0.453	0.350	0.394	0.472	0.354	0.405	0.507	0.348	0.413	0.485	0.323	0.388
	400/50	0.450	0.349	0.393	0.468	0.354	0.403	0.503	0.350	0.412	0.509	0.328	0.399
	600/25	0.419	0.344	0.378	0.439	0.352	0.391	0.492	0.364	0.419	0.522	**0.365**	**0.430**
	600/50	0.419	0.343	0.377	0.444	0.354	0.394	0.485	0.353	0.408	**0.532**	0.353	0.425
	800/20	0.416	0.344	0.377	0.431	0.348	0.385	0.493	0.351	0.410	0.513	0.343	0.411
	800/50	0.419	0.345	0.378	0.433	0.350	0.387	0.515	0.346	0.414	0.517	0.338	0.409
AImed (1 class)	200/25	0.405	0.545	0.465	0.430	0.509	0.466	0.480	0.484	0.482	0.507	0.460	0.482
	200/50	0.405	0.545	0.465	0.430	0.509	0.466	0.480	0.484	0.482	0.507	0.460	0.482
	400/25	0.420	0.589	0.491	0.451	0.554	0.497	0.481	0.524	0.501	0.516	0.502	0.509
	400/50	0.420	0.588	0.490	0.455	0.561	0.502	0.484	0.529	0.505	**0.519**	0.505	0.512
	600/25	0.409	0.605	0.488	0.445	0.571	0.500	0.475	0.529	0.500	0.511	0.513	0.512
	600/50	0.409	0.605	0.488	0.445	0.571	0.500	0.475	0.530	0.501	0.511	0.513	0.512
	800/25	0.416	0.613	0.496	0.453	0.595	0.514	0.481	0.547	0.512	0.490	0.512	0.501
	800/50	0.418	**0.614**	0.498	0.454	0.596	**0.515**	0.482	0.545	0.511	0.489	0.514	0.501

Table 3: Results obtained with different configurations of the proposed method.

Finally, the AImed corpus[5] consists of 225 MEDLINE abstracts, 200 of which describing interactions between human proteins. There are 4,084 protein references and approximately 1,000 tagged interactions. In this data set there is no distinction between genes and proteins, and the relations are symmetric and of a single type. We relied on a 10-fold cross validation methodology in the experiments with AImed, using the same splits as Mooney and Bunescu [4].

Using the three different datasets, we experimented with different parameters for the minwise hashing-based scheme, namely by varying the number of nearest neighbors that was considered (1, 3, 5 or 7), the size of the min-hash signatures (200, 400, 600 or 800 integers) and the number of LSH bands (25 or 50 bands). Notice that when using b bands of r rows each, the probability that the signatures of two sets S_1 and S_2 agree in all the rows of at least one band, therefore becoming a candidate pair, is $1-(1-\mathrm{J}(S_1,S_2)^r)^b$. With 50 bands and a min-hash signature of size 600, roughly one in 1000 pairs that are as high as 85% similar will fail to become a candidate pair through the LSH method, and thus be a false negative. With these parameters, instances with a similarity bellow 85% are also likely to be discarded through the LSH method, which can contribute to the confidence in a correct classification (i.e., we are also trading precision for recall, when selecting the minwise hashing parameters).

As evaluation measures, we used macro-averaged precision, recall and F1-scores over the relation labels, apart from the *not-related/other* labels. This corresponds to calculating macro-averaged scores over 18 classes in the case of the SemEval dataset, over 46 classes in the case of the Wikipedia dataset, and we

[5] `ftp://ftp.cs.utexas.edu/pub/mooney/bio-data/`

measured results over the single *is-related* class in the AImed dataset. Table 3 presents the obtained results, showing that using the 5 or the 7 nearest neighbors, instead of just the most similar example, results in an increased performance for the SemEval and Wikipedia datasets, while a better F1 score was obtained for the AImed dataset when considering the 3 nearest training examples.

Table 4 presents per-class results in the case of the SemEval dataset, considering the configuration that achieved the best performance in the results from Table 3 (i.e., a configuration using the 5 nearest neighbors, with a min-hash size of 400, and with 50 bands in the LSH method). Besides the results on the regular SemEval classification setting, involving relation types with direction, we also present results in a setting that ignores the relation directions (i.e., considering 8 different relationship classes) as well as individual results for the class corresponding to *other-relations*. The results show that some classes, such as *cause-effect* are relatively easy to classify, whereas classes such as *instrument-agency* are much harder. For the class corresponding to *entity-destination(e1,e2)*, the dataset only contains one instance for training and one instance for testing.

In terms of comparisons with the current state-of-the-art, the best participating system at the SemEval 2010 task achieved a performance of over 0.82 in terms of the F1 metric, whereas the second best system reported an F1 score of 0.77. The median F1 score was of 0.68, while our best F1 score was of 0.69. Participating systems used a variety of methods and resources. For instance the winning entry used Google's n-gram collection and an approach that splits the task into two classification steps (i.e., relationship type and direction), whereas the second best participant used Cyc as a semantic resource. Almost all participants used features derived from WordNet, Roget's Taxonomy, or Levin's verb classes. In our approach, we used a simpler set of features common to different domains and mostly language independent: we only need POS tags for computing some of the features, and POS tagging can be made efficiently and accurately for most languages [14]. Comparing with other approaches over the same datasets, our method is focused on addressing scalability, although still attaining competitive results in terms of accuracy.

In the specific case of the AImed dataset, previous studies have mostly compared different kernel-based methods, reporting F1 scores ranging from 0.26 to 0.60, with a common cross-validation methodology [1,17]. For instance the subsequence kernel from Bunescu and Mooney achieves an F1 score of 0.54 [4], while the all-dependency-paths kernel from Airola et al. achieves an F1 score of 0.56 [1]. Our min-hash approach has slightly inferior results, with an F1 score of 0.52, but we argue that our method has advantages over kernel-approaches in terms of simplicity, support for multi-class on-line learning, and scalability.

Our approach is significantly different from that of commonly used kernel-based methods, which involve two main components. First, the linguistic structures, to be used within the kernels, have to be generated. Dependency parsers can be about an order of magnitude faster than syntax parsers, on average taking 130 milliseconds per sentence on modern hardware. POS tagging is about 1.5 orders of magnitude faster than dependency parsing, on average taking 4

Relation	Direction	Instances (train/test)	Asymmetrical			Symmetrical		
			Precision	Recall	F1	Precision	Recall	F1
Cause-Effect	(e1,e2)	344/134	0.843	0.843	0.843	0.798	0.902	0.847
	(e2,e1)	659/194	0.735	0.902	0.810			
Component-Whole	(e1,e2)	470/162	0.572	0.759	0.653	0.628	0.670	0.648
	(e2,e1)	150/129	0.609	0.520	0.561			
Entity-Destination	(e1,e2)	844/291	0.744	0.911	0.819	0.747	0.901	0.817
	(e2,e1)	1/1	1.000	0.000	0.000			
Entity-Origin	(e1,e2)	568/211	0.789	0.815	0.802	0.756	0.795	0.775
	(e2,e1)	148/47	0.667	0.723	0.694			
Product-Producer	(e1,e2)	323/108	0.670	0.602	0.634	0.673	0.589	0.628
	(e2,e1)	394/123	0.654	0.569	0.609			
Member-Collection	(e1,e2)	78/32	0.778	0.438	0.560	0.767	0.777	0.772
	(e2,e1)	612/201	0.776	0.791	0.783			
Message-Topic	(e1,e2)	490/210	0.751	0.733	0.742	0.778	0.778	0.778
	(e2,e1)	144/51	0.750	0.706	0.727			
Content-Container	(e1,e2)	374/153	0.726	0.778	0.751	0.706	0.802	0.751
	(e2,e1)	166/39	0.627	0.821	0.711			
Instrument-Agency	(e1,e2)	97/22	0.429	0.545	0.480	0.605	0.667	0.634
	(e2,e1)	407/134	0.615	0.679	0.645			
Other	—	1410/454	—	—	—	0.442	0.293	0.352
Macro-average	—	—	0.708	0.674	0.690	0.718	0.764	0.740

Table 4: Results obtained for different relation classes in the SemEval dataset.

milliseconds per sentence on modern hardware. Our method mostly relies on character quadgrams, although we also used POS tags for improving accuracy. Nonetheless, we avoid complex NLP operations associated with parsing the sentences. Second, we have that the substructures used by the kernels have to be determined, and the classifier has to be applied. Using the AImed dataset, Tikk et al. reported on times of approximately 66.4 and 10.8 seconds, for training and testing (i.e., without feature extraction) an SVM classifier using a shallow linguistic kernel that is essentially a simplified version of the subsequence kernel from Bunescu and Mooney [4], as well as times of approximately 4517.4 and 3.7 seconds from training and testing, using the all-paths-graph kernel [17]. In our experiments, which were executed on modern hardware and using a single core, it took us 172 seconds for processing all three stages (i.e., feature extraction, training and testing) with the SemEval 2010 dataset, when considering the 5 nearest neighbors, min-hash signatures of size 400, and 50 bands. Each fold from the AImed dataset takes around 161 seconds to process considering the 3 nearest neighbors, min-hash signatures of size 800 and 50 bands.

Although a direct comparison against the current stat-of-the-art methods, in terms of computational performance, cannot be made easily (i.e., this would require a common set of tools for performing feature extraction, as well as running the implementations of the different algorithms on the same hardware and over exactly the same datasets) we provide some detailed figures for the performance of our method. The charts on Figure 1 present the processing times, in seconds, that are separately involved in feature extraction from the training and testing data, training (i.e., data indexing) and classification, for the different settings represented in Table 3. In the case of the AImed dataset, the charts show the average time over the 10 different folds.

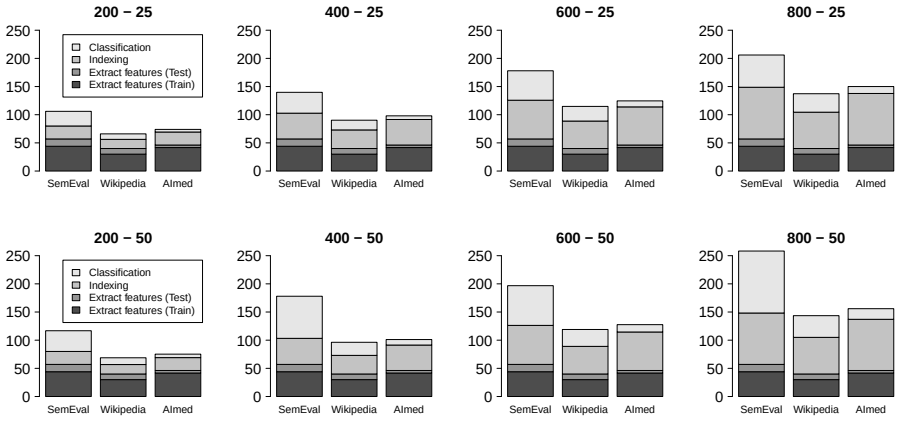
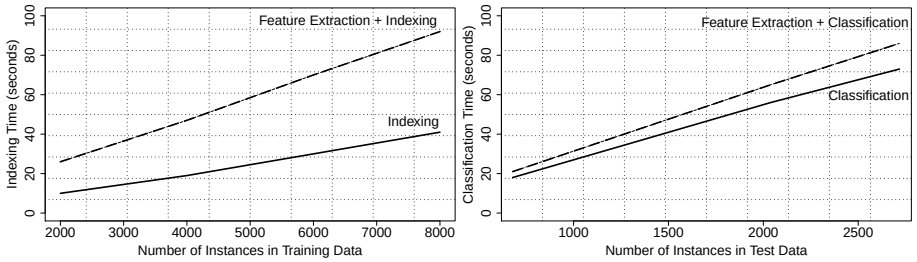


Fig. 1: Processing times, in seconds, for each dataset and configuration.

The total processing time involved in each experiment naturally increases with the size of the dataset being considered. The time taken to extract features is independent of the LSH configuration being used, and the results indicate that these values represent a significant amount of the total processing time that is involved in each experiment. The results also show that the indexing times increase significantly as the size of the min-hash signatures gets larger, since more hash functions need to be computed, and more min-hash values have to be stored and compared. Augmenting the number of bands in turn increases the classification time, since the number of hash tables where we have to look for candidate instances, and possibly also the number of candidates increases.

Figure 2a shows how the proposed method performs, in terms of processing times, over the training phase and with an increasingly larger dataset, specifically when considering 25%, 50%, 75% and 100% of the SemEval dataset. In Figure 2b, the training was performed over the full SemEval training dataset, and then the classification time was also measured for different partitions of the test dataset. Both times were measured with the configuration that achieved the best performance on SemEval, in terms of the F1 score – see Table 3. The time taken to process the data, both for training and for classification, grows almost linearly with the dataset size.

Besides simplicity, computational efficiency, and direct support for multi-class classification, our method also has the advantage of being completely on-line, since to consider new training examples, we only need to compute their min-hash signatures and store them in the LSH hash tables.



(a) Indexing Time (b) Classification Time

Fig. 2: Processing times with different partitions of the SemEval dataset.

5 Conclusions and Future Work

Through this work, we have empirically evaluated a min-hash based method for fast extraction of semantic relationships from textual documents. We made experiments with well known datasets from three different application domains, showing that the task can be performed with high accuracy, using a simple on-line method based on similarity search, that is also computationally efficient.

Despite the interesting results, there are also opportunities for improvement. For instance, some of the state-of-the-art kernel methods for relation extraction also explore similarity between graph-based representations for the relation instances, derived from both lexical information and from constituency/dependency parsing [13]. Recent studies have proposed minwise hashing methods for comparing graphs [16], and it would be interesting to experiment with the application of these methods in the task of relationship extraction from textual documents, using richer graph-based representations for the relation instances. The proposed approach for measuring similarity between relation instances, based on the Jaccard similarity coefficient, could also be integrated into a general kernel-based framework (i.e., an SVM classifier using a kernel function based on the Jaccard similarity coefficient), and it would be interesting to see if supervised learning methods could be used to discover weights for properly combining different Jaccard similarity coefficients, computed for instance with basis on different sub-strings surrounding the relation operands.

Since the seminal work of Broder on minwise hashing [2], there have also been considerable theoretical and methodological developments in terms of the application of minwise hashing techniques. For future work, we would like to experiment with the b-bit minwise hashing approach presented by Li and König [12] for improving storage efficiency on very large datasets, or with the extension proposed by Chum et al. for approximating weighted set similarity measures [5].

Acknowledgements

This work was partially supported by FCT, through project grants PTDC/EIA-EIA/109840/2009 (SInteliGIS) and UTA-Est/MAI/0006/2009 (REACTION), as well as through the INESC-ID multi-annual funding from the PIDDAC programme (PEst-OE/EEI/LA0021/2013). The author David Batista was also supported through the PhD scholarship from FCT with reference SFRH/BD/70478/2010.

References

1. A. Airola, S. Pyysalo, J. Björne, T. Pahikkala, F. Ginter, and T. Salakoski. A graph kernel for protein-protein interaction extraction. In *Proceedings of the Workshop on Current Trends in Biomedical Natural Language Processing*, 2008.
2. A. Broder. On the resemblance and containment of documents. In *Proceedings of the Conference on Compression and Complexity of Sequences*, 1997.

3. R. Bunescu and R. Mooney. A shortest path dependency kernel for relation extraction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2005.
4. R. Bunescu and R. Mooney. Subsequence kernels for relation extraction. In *Proceedings of the Conference on Neural Information Processing Systems*, 2006.
5. O. Chum, J. Philbin, and A. Zisserman. Near duplicate image detection: min-hash and tf-idf weighting. In *Proceedings of the British Machine Vision Conference*, 2008.
6. A. Culotta, A. McCallum, and J. Betz. Integrating probabilistic extraction models and data mining to discover relations and patterns in text. In *Proceedings of the Conference of the North American Chapter of the ACL*, 2006.
7. A. Culotta and J. Sorensen. Dependency tree kernels for relation extraction. In *Proceedings of the Annual Meeting of the ACL*, 2004.
8. A. Fader, S. Soderland, and O. Etzioni. Identifying relations for open information extraction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2011.
9. I. Hendrickx, N. Kim, Z. Kozareva, P. Nakov, D. Séaghdha, S. Padó, M. Pennacchiotti, L. Romano, and S. Szpakowicz. Semeval-2010 task 8: Multi-way classification of semantic relations between pairs of nominals. In *Proceedings of the International Workshop on Semantic Evaluation*, 2010.
10. N. Kambhatla. Combining lexical, syntactic, and semantic features with maximum entropy models for extracting relations. In *Proceedings of the Annual Meeting of the ACL*, 2004.
11. S. Kim, J. Yoon, J. Yang, and S. Park. Walk-weighted subsequence kernels for protein-protein interaction extraction. *BMC Bioinformatics*, 11(107), 2010.
12. P. Li and C. König. b-bit minwise hashing. In *Proceedings of the International Conference on World Wide Web*, 2010.
13. T.-V. Nguyen, A. Moschitti, and G. Riccardi. Convolution kernels on constituent, dependency and sequential structures for relation extraction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2009.
14. S. Petrov, D. Das, and R. T. McDonald. A universal part-of-speech tagset. In *Proceedings of the Conference on Language Resources and Evaluation*, 2012.
15. A. Rajaraman and J. Ullman. *Mining of massive datasets*, chapter 3. Finding Similar Items. Cambridge University Press, 2011.
16. C. Teixeira, A. Silva, and W. Junior. Min-hash fingerprints for graph kernels: A trade-off among accuracy, efficiency, and compression. *Journal of Information and Data Management*, 3(3), 2012.
17. D. Tikk, P. Thomas, P. Palaga, J. Hakenberg, and U. Leser. A comprehensive benchmark of kernel methods to extract protein–protein interactions from literature. *PLoS Computational Biology*, 6(7), 2010.
18. D. Zelenko, C. Aone, and A. Richardella. Kernel methods for relation extraction. *Journal of Machine Learning Research*, 3, 2003.
19. Y. Zhang, H. Lin, Z. Yang, J. Wang, and Y. Li. Hash subgraph pairwise kernel for protein-protein interaction extraction. *IEEE/ACM Transactions on Computer Biology and Bioinformatics*, 9(4), 2012.
20. S. Zhao and R. Grishman. Extracting relations with integrated information using kernel methods. In *Proceedings of the Annual Meeting of the ACL*, 2005.
21. G. Zhou and M. Zhang. Extracting relation information from text documents by exploring various types of knowledge. *Information Processing and Management*, 43(4), 2007.