Relation extraction methods for biomedical literature

Bui, Q.C.

Chapter 2. Relation extraction methods for biomedical text

Abstract

This chapter provides a background and a review of existing techniques for extracting relations from biomedical text. We start with an overview of a relation extraction system. We then focus on each component employed in the extraction system, namely name entity recognition (NER), NLP tools, and ML methods. For each component being discussed, we provide a list of popular tools that are available for use. Furthermore, we also discuss the role of annotated corpora, how they influence the selection extraction methods. Finally, we present main approaches and their state-of-the-art results of relation extraction from biomedical text.

2.1 Introduction

With the huge amount of information hidden in biomedical text, which is measured by the numbers of publications, that is increasing with an exponential rate, it is no longer possible for a researcher to keep up-to-date with all developments in a specific field [3, 6]. In addition, the focus of biomedical research shifts from individual genes or proteins to entire biological systems, making the demand for extracting relationships between biological entities (e.g. protein-protein interactions, genes-diseases) from biomedical text to discover knowledge and to generate scientific hypotheses increasingly urgent [7, 8]. Manually transforming this information from unstructured text into a structured form is a time-consuming and laborious task [24]. Automatic relation extraction offers an interesting solution to this problem because it can reduce time spent by researchers on reviewing the literature and by significantly covering many more scientific articles than those normally reviewed [38]. Therefore, many approaches have been proposed to extract relations from biomedical text in recent years, ranging from simple co-occurrence approaches to sophisticated machine learning-based approaches. These approaches differ from each other in many respects such as how the natural language processing (NLP) techniques are used to analyze the input text and which methods are used to learn extraction rules i.e. manually defined pattern or automatically learn from training datasets [28]. In general, a relation extraction system consists of three modules, namely text preprocessing, parsing, and relation extraction as shown in Figure 2.1.

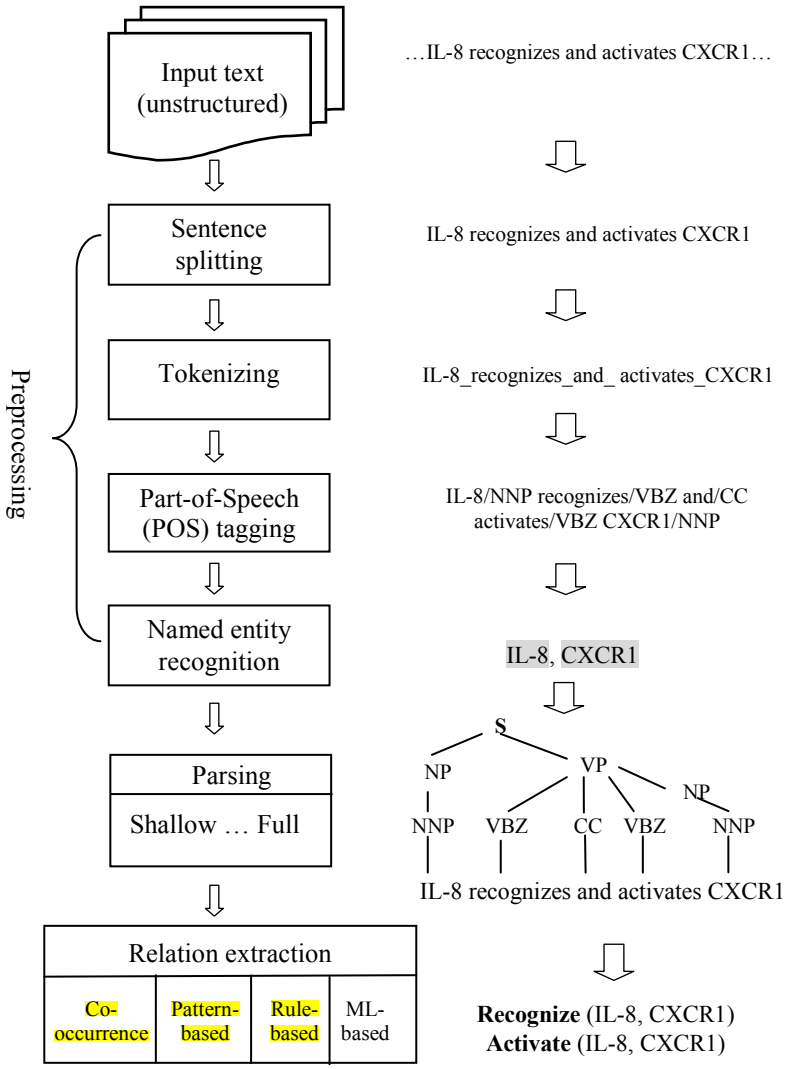

Figure 2.1. Workflow of a typical relation extraction system

2.1.1 Text preprocessing

Text preprocessing module splits documents into basic text units for subsequent processing. Currently, most of relation extraction systems work with sentence-based unit therefore larger blocks of text such as abstracts, paragraphs or whole documents are split into single sentences. Based on this unit, a tokenization step is carried out to further split the sentence into tokens, which are then used as input for subsequent processes such as the named entity recognition step. The sentence splitting,

tokenizing, and part-of-speech tagging steps are described in more details in the NLP section.

2.1.2 Named entity recognition

The aim of biomedical named entity recognition (NER) is to identify the biomedical entities (e.g. names of genes and proteins) mentioned in text. NER is a prerequisite step for any relation extraction systems [2]. Due to the complexities of the biomedical terminology, recognizing named entities in biomedical text is considered as a daunting task [39]. The main issue in NER is the lack of standardization of names. First, each biomedical entity such as gene or protein might have several names and abbreviations, for example, gene *BRCA1* is also known as *breast cancer 1*, or *Brca1*, *BRCA 1*, and *brca1*, etc. Second, the same name can refer to different entities depending on the context e.g. *Cdc2* can be two completely unrelated genes. Third, biomedical entities may have multi-word names which cause overlap of candidate names. A recent study by [34] has shown that the performance of the same PPI extraction system, which is measured in terms of F-score, can drop 15% when applied to two identical datasets, in which, one dataset is given the gold standard protein names and the other uses a NER tool to recognize protein names.

To recognize these names, the early NER methods rely on manually defined rules which based on the characteristics of names such as morphological and syntactic features. As annotated corpora are now available, more NER systems are based on machine learning methods [40, 41]. To reduce the number of false positives, some systems also rely on dictionaries which consist of a list of synonyms of entities [42]. Currently, the performance of the state-of-the-art NER systems for biomedical texts in terms of F-scores varies from 78% to 90%. A list of available NER tools is shown in Table 2.1.

Table 2.1. List of common NER tools for biomedical text

Name	Description	URL
BANNER	Trained on BioCreative 2 GM data.	http://banner.sourceforge.net
Lingpipe	Trained on GENIA corpus.	http://alias-i.com/lingpipe/
GENIA Tagger	Trained on GENIA corpus.	http://www.nactem.ac.uk/tsujii/GENIA/tagger/
ACELA	Trained on GENIA.	http://www.nactem.ac.uk/acela/

2.1.3 Parsing

The goal of the parsing step is to transform unstructured text into more structured forms, which may reveal linguistic patterns or common similarities of the written text. Based on the parsing output, methods are built to extract relations between biomedical entities. Depending on the extraction techniques, the use of parsing tools to analyze text vary from shallow parsing to dependency parsing and deep parsing. The details of these parsing techniques are described in section 2.2

2.1.4 Relation extraction

Relation extraction is the core module of a relation extraction system. Many techniques have been proposed to extract relation from biomedical text. Early systems focus on extracting a small set of simple relations such as inhibit or binding relations between proteins while recent systems attempt to extract more complex relations such as protein-protein interactions or biomedical events. In general, techniques used in relation extraction systems can be divided into four groups, namely co-occurrence, pattern-based, rule-based, and machine learning-based approaches. The detail of each approach is described in section 2.5.

2.1.5 Evaluation metrics

To evaluate the performance of a relation extraction system, the following metrics are used: recall, precision, and the F-score.

Recall = TP/(TP + FN)

Precision = TP/(TP + FP)

F-score = 2 * Recall * Precision/(Recall + Precision),

where TP, FN, and FP are defined as:

TP (true positives): is the number of relations that were correctly extracted from input documents.

FN (false negatives): is the number of relations that the system failed to extract from input documents.

FP (false positives): is the number of relations that were incorrectly extracted from input documents.

The F-score is the harmonic mean of recall and precision.

The remaining structure of this chapter is as follows. In section 2.2, we introduce common NLP techniques used in relation extraction methods. We discuss the use of machine learning methods for relation extraction in section 2.3. We then introduce the biomedical corpora and evaluation metrics in section 2.4. In section 2.5 we present the current approaches to relation extraction.

2.2 Natural Language Processing

Natural language processing (NLP) is the process of analyzing text in natural language aiming to convert unstructured text into a structured form. Because of its complexity, the analysis of the natural language is carried out in many steps, which are: lexical, syntactic, and semantic analyses [25]. Lexical analysis is the process of converting a sentence into a sequence of tokens. Once tokens are determined, morphological analysis is carried out to map lexical variants of a word into its canonical base form. Syntactic analysis consists of the assignment of part-of-speech (POS) tags (.e.g., noun, verb, adjective, etc.) to the sequence of tokens that makes up a sentence and determining the structure of a sentence through parsing tools. Semantic analysis determines the meaning of a sentence and anaphora resolution. In this thesis we only focus on NLP techniques that are frequently used in relation extraction systems which are lexical and syntactic analyses. Currently, NLP techniques can only handle input text at the sentence level therefore sentence splitting is usually the first step in a NLP pipeline.

2.2.1 Sentence splitting

Sentence splitting is the process of determining sentence boundaries. It splits input text such as abstracts and paragraphs into single sentences and is a prerequisite step for subsequent processes [43]. Although it seems to be straightforward, sentence splitting is a non-trivial task. There are many issues that cause this task to become difficult, such as the irregularity in proteins or genes names (e.g. E. coli, p53), inline citations (e.g., Proc. Acad. Sci. 2006), and abbreviations (e.g. Dr., et al.). To improve the accuracy of the splitting methods over biomedical text, most of sentence splitting tools are necessary to re-train on biomedical corpora. Available tools such as Ling-Pipe can achieve an accuracy of 98% when evaluating on biomedical test sets.

2.2.2 Lexical processing

This process deals with how input text is split into words (tokens) and how *base* forms of a word are found.

Tokenization

Tokenization is the process of splitting the input sentence into a sequence of tokens. This is a prerequisite step before any linguistic analysis can be carried out. Errors that occur in this step propagate to the other levels and thus deteriorate the performance of subsequent tasks such as NER and relation extraction [44]. The simplest way to tokenize input text is based on white spaces and punctuation symbols. Similar to the sentence splitting process, the tokenization process encounters many problems such as abbreviation, in which words are not always separated from other tokens by white space. This abbreviation problem may also

interfere with the sentence splitting process above. Another problem is the use of hyphenation since it is ambiguous to determine whether to return one or more tokens for hyphenated words. For example, Cdc2-depedent should be treated as two tokens, whereas DNA-binding should return as a single token. Moreover, in the biomedical domain, the tokenization process has also to deal with additional challenges due to domain-specific terminology, nonstandard punctuation, and orthographic patterns e.g., 12.0 +/- 1.6%, and *alpha-galactosyl-1,4-beta-galactosyl-1,4-glucosyl*. Currently, tokenization tools for biomedical text achieve an accuracy of 95%. A list of available tokenization tools is shown in Table 2.2.

Stemming

Stemming (morphological analysis) is the process of mapping various syntactic forms of a word into its canonical base form. The purpose of this process is to reduce the variation among tokens, which reduce the sparseness for lexical representation of the text [45]. For example, one can represent all morphological variants of relations between two entities A and B such as: activation of A and B, A activates B, A activated B, and A activating B, by a single relation activat(A,B).

Stemming is a standard technique which is commonly used to create the bags-of-words (BOW) features in many relation extraction systems. This tool is usually bundled with the NLP packages.

2.2.3 Syntactic processing

Syntactic processing is an important part of a NLP pipeline as this process reveals structural relationship between groups of words at the sentence level. This process consists of three stages: part-of-speech tagging, chunking and full parsing.

POS tagging

POS tagging is the first step of syntactic analysis and is essential to cope with the various lexico-syntactic ambiguity forms of words. For example, the word *result* can be either a common noun or a verb, depending on its syntactic context. The POS tagger receives input as a sequence of words of a sentence, and assigns POS tags to the words of that sentence. It identifies the grammatical form of a word based on the word itself and its surrounding context. Such grammatical forms output from the POS tagger are nouns, verbs, preposition, etc., and are categorized into 8 basic groups.

Recently, most of POS taggers are built based on ML methods [46]. These methods require training data which are words with manually assigned POS tags. POS taggers which are trained on wire news corpora such as the Wall Street Journal corpus can achieve an accuracy of 98% on general texts. However, they need to be re-trained on biomedical corpora in order to achieve the same accuracy level when

applied to biomedical texts [47]. A list of available POS taggers which are commonly used in relation extraction system is shown in Table 2.2.

In relation extraction systems, output of a POS tagger are used to form patterns for pattern-based systems or used as features for ML-based systems. An example of POS tags is shown in Figure 2.1. The use of POS tags is discussed in section 2.5

Shallow parsing

Shallow parsing is the process of combining sequences of words into syntactic groups such as noun and verb phrases using both lexical and POS sequence information. It is a preliminary stage to help fully parsing a sentence. In shallow parsing, the structure of a sentence is not resolved to the level of single elements. Similar to POS tagging, most of modern shallow parsers are relied on the availability of training corpora which are annotated with chunks. To achieve high accuracy on biomedical text, retraining these shallow parsers on annotated biomedical corpora is essential. Currently, the performance of the state-of-the- art shallow parsers achieve an accuracy of 96% when evaluated on biomedical test corpora [27]. A list of available shallow parser is shown in Table 2.2. An output of shallow parsing is shown in Figure 2.2a.

Many relation extraction systems employ shallow parsing to analyze input text. These systems range from rule-based to ML-based approach. The detailed usage of the shallow parsing output is discussed in section 2.5.

Full parsing

Full parsing is the process of analyzing syntactic structure of a sentence with the most elaborated details. It accumulates the output of all previous steps such as POS tags, phrases (chunks) and adds more information about structural dependencies between phrases. The full parsing output of a given sentence is a parse tree which reveals the relationship of subject-predicate-object of that sentence. Full parsing techniques can be divided into three types: phrase structure parsing, dependency parsing, and deep parsing [26]. Phrase structure parsing produces a constituent tree of a sentence, where syntactic tags are inner nodes and words are leaves, as shown in Fig 3. Dependency parsing produces tree structure of a sentence, where nodes are words and edges represent the relationships among words, as shown in Figure 2.4. Deep parsing produces theory-specific syntactic and semantic structures and predicate argument structures which represents the relationship among words [48]. With the availability of biomedical treebanks, existing general purpose parsers which are trained on general text such as the WSJ corpus are now able to retrain on biomedical domain. The performance of the state-of-the-art full parser for biomedical text achieves an accuracy of 80%. A list of available full parser is shown in Table 2. Examples of full parsing are shown in Figure 2.1 and Figure 2.2.

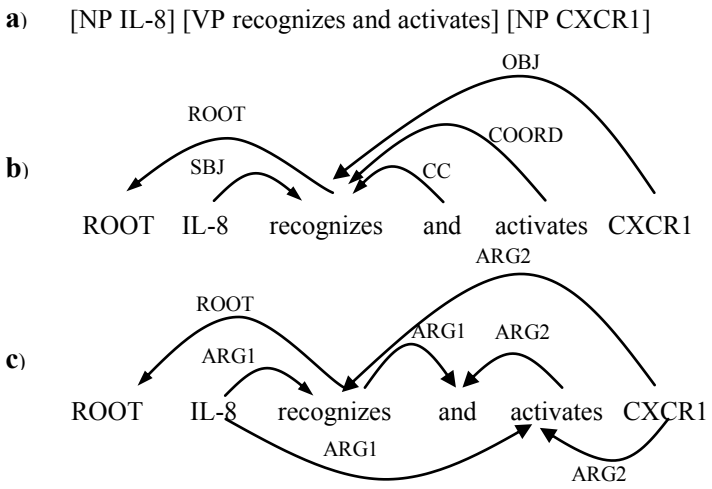

Figure 2.2. Parsers output of sentence "IL-8 recognizes and activates CXCR1". (a) output of the shallow parser. (b) output of the dependency parser. (c) output of the deep parser in form of predicate argument structure.

The use of full parsing in relation extraction systems vary depending on the parsing types as well as extraction techniques. The detailed usage of the full parsing output is discussed in section 2.5.

2.3 Machine learning

Machine learning (ML) techniques are widely used as a component of relation extraction methods. In this section we introduce the basic concepts of the ML techniques and explain how they are used in the relation extraction systems. The details of general ML algorithms and specific algorithms for relation extraction can be found in [49] and [33], respectively.

Machine learning methods are based on statistical analysis of data to infer general rules. It stems from the situation that we do not explicitly know solutions to solve the problems but data related to these problems are available. In relation extraction, such data can be annotated text, sentences, or phases containing relations between entities. The task of a ML method is either to learn rules from these examples which reveal the structure of the underlying data, or to distinguish instances of data from each other. Therefore, the outcome of a ML method is either the learning rules or a model which is used to predict unknown data based on previous seen data [49]. For example, given a set of biomedical data containing gene-disease associations, a ML method then learns a model to predict gene-disease associations from unseen biomedical data. In this case, the learning method corresponds to the supervised machine learning paradigm, which consists of two

phases: training and testing phase. In the training phase, ML methods build a model by learning sets of properties and their respective values of the examples from the training data. Such properties are called features. It is based on the observation that examples belongs to the same class (e.g. interaction, non-interaction) have more features in common than examples that belong to other classes. Such features are then used by ML methods to decide whether an example belongs to a specific class or not. Therefore, most ML algorithms try to find a set of features, their values, and combinations of features to distinguish all classes from each other. Although many ML methods have been proposed, the most commonly used ML methods for relation extraction are support vector machines (SVM) [50].

Table 2.2. A list of common NLP tools for biomedical text.

Category	Name	URL
Sentence splitter	Lingpipe	http://alias-i.com/lingpipe
	Enju	http://www.nactem.ac.uk/y-matsu/geniass
Tokenization	OpenNLP	http://opennlp.apache.org
	Stanford	http://nlp.stanford.edu/software/tokenizer.shtml
	Lingpipe	http://alias-i.com/lingpipe
POS tagger	Enju	http://www.nactem.ac.uk/GENIA/tagger/
	OpenNLP	http://opennlp.apache.org/
	Stanford	http://nlp.stanford.edu/software/tokenizer.shtml
	Lingpipe	http://alias-i.com/lingpipe/
Shallow parser	OpenNLP	http://opennlp.apache.org/
	Illinois Chunker	http://cogcomp.cs.illinois.edu/page/software
	Lingpipe	http://alias-i.com/lingpipe/
Full parser	Stanford	http://nlp.stanford.edu/software/lex-parser.shtml
	Charniak–Lease re-ranking	ftp://ftp.cs.brown.edu/pub/nlparser/reranking-parserAug06.tar.gz
	OpenNLP	http://opennlp.apache.org/
	Enju	http://www.nactem.ac.uk/enju/

2.3.1 SVM methods for relation extraction

Most relation extraction systems use ML methods as the classifiers. In a nutshell, these methods work as follows. Given a training set, for example, a list of annotated sentences, some of which contain PPI pairs (positive examples) and some contain non-interacting protein pairs (negative examples). All sentences are transformed into representations such that ML methods can capture properties or features that are best expressed the interaction pairs (or not for negative examples). The simplest representation form is a list of words that occur in the sentences. More complex representations are parse trees obtained from the output of NLP tools which can reveal the structure and dependencies of words in the sentence. The set of structured representations and the PPIs are then used as input for a ML classifier i.e. SVM classifier to learn a model of how PPIs are typically expressed. To predict new PPI pairs from unseen text, every new sentence must be transformed into the same representation as the training sentences, the SVM classifier then use the learned model to classify whether each candidate PPI pair is an interaction or not. The mechanism for a SVM classifier to carry out such classification is to find a hyperplane that can separate positive and negative examples with the largest possible margin [36, 37]. Such a hyperplane is learned from training examples. Training examples that lie closest to the hyperplane are the support vectors. To classify a new object, the SVM classifier checks on which side of the hyperplane that object resides. Imagination that these examples can be linearly separated, e.g. in two dimensional case, then we can draw a line that separates two classes as shown in Figure 2.3. A list of features that commonly used in relation extraction is described in section 2.5.

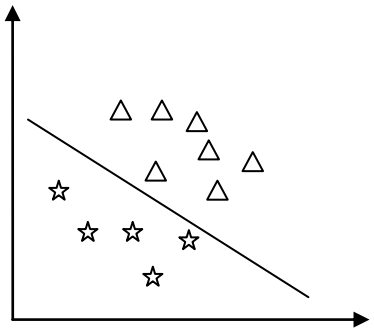

Figure 2.3. A hyperplane which separates the class of instances represented by stars from the class of instances represented by triangles.

2.3.2 Kernel methods

There are many cases where the classes in data sets are not linearly separable. In these cases, training data can be mapped into higher dimensional space through a non-linear mapping. In the new space, the classes may be separable and the hyperplane classifier can be applied. This non-linear mapping is done via a kernel function which takes the representation of two examples and computes their similarity [51].

Many kernels have been proposed, ranging from general purpose to user-defined kernels. While general kernels are applicable to arbitrary problems, user-defined kernels incorporate domain knowledge to fit specific applications. In relation extraction, user-defined kernels mainly differ from each other by their structural representations and how the similarity functions are calculated [33]. Some common kernels are discussed in section 2.5

2.4 Biomedical corpora

Biomedical corpora play an important role in the development of relation extraction methods, such as providing data to retrain existing NLP tools to work with biomedical texts, to train ML-based relation extraction approaches, and to facilitate automatic performance evaluation of extraction systems. Since creating biomedical corpora is a time-consuming and error prone task, the number of available biomedical corpora is small. A full list of the available biomedical corpora can be found in the WBI corpus repository: http://corpora.informatik.hu-berlin.de. Some typical corpora are listed bellows:

2.4.1 GENIA corpus

GENIA is the largest annotated corpus publicly available in the biomedical domain [52]. It consists of 2,000 Medline abstracts with more than 400,000 words and almost 100,000 annotations for biological terms. In addition, it contains annotation for linguistic structures such as part-of-speech, phrasal and syntactic structures. This corpus is often used to retrain NLP tools such as tokenizers, POS taggers, and full parsers to work with biomedical text.

2.4.2 PPI corpora

There is a small set of five PPI corpora, namely AIMed [53], BioInfer [54], HPRD50 [55], IEPA [56], and LLL [57]. These corpora are created independently for different purposes. Although these corpora contain annotation for named entities and PPI pairs, they vary in many aspects including size (e.g. number of abstracts ranging from 50 to 200), biological coverage (i.e. retrieved from PubMed using different keywords) and annotation policy (e.g. interaction type, direction of interactions). To minimize the differences between these corpora, [29] have

transformed these PPI corpora into an XML-based format and proposed to use only undirected and untyped PPI pairs from these corpora for evaluation purposes. With the introduction of these unified PPI corpora, it is easier to compare performance between PPI extraction methods. Furthermore, new evaluation methods based on these PPI corpora have also been proposed which are cross-learning and cross-corpora [58]. In cross-learning, four corpora are used for training and the fifth corpus is used for testing, whereas in cross-corpora, one corpus is used for training and the other four are used for testing. These evaluations reveal the ability of an extraction method adapting to new text with unknown characteristics. This is one step closer to the real world situation where the input text is diversity. These PPI corpora are used to evaluate our method to extract PPI in Chapter 4.

2.4.3 GENIA events corpus

The GENIA *events* corpus is based on the GENIA corpus, consisting of 1,000 Medline abstracts [59]. It contains 9,372 sentences in which 36,114 events are identified. An event consists of a trigger, type and participants of the event. The event trigger is a word or phrase in the sentence which indicates the occurrence of the event. The event type categorizes the type of information expressed by the event. The event participants are entities or other events. Table 2.3 shows a list of defined events and their arguments. This corpus and 15 annotated full-text documents are used to evaluate our method to extract biological events in Chapter 5.

Table 2.3. Event types and their arguments for Genia event extraction task [60]. Secondary arguments are optional. P denotes for Protein, Ev denotes for event

Type	Primary Argument	Secondary Argument
Gene_expression	Theme(Protein)	
Transcription	Theme(Protein)	
Protein_catabolism	Theme(Protein)	
Phosphorylation	Theme(Protein)	Site
Localization	Theme(Protein)	AtLoc,ToLoc
Binding	Theme(Protein)+	Site+
Regulation	Theme(P/Ev),Cause(P/Ev)	Site, CauseSite
Positive_regulation	Theme(P/Ev),Cause(P/Ev)	Site, CauseSite
Negative_regulation	Theme(P/Ev),Cause(P/Ev)	Site,CauseSite

2.5 Relation extraction methods

Various approaches have been proposed to extract relations from biomedical text [58, 61–66]. Due to the inherent complexity of biomedical text, most of relation extraction systems work on sentence-based level. The approaches used in relation extraction systems vary from the level of linguistic analysis to the way patterns or rules are being learned. Based on the techniques employed in these systems, we can categorize them into four groups, namely co-occurrence, pattern-based, rule-based, and ML-based approaches. In the following sections we present the most common characteristics of these methods (while ignoring their targeted relation types). The detailed methods to extract the specific type of relations such as PPIs and biological events are left in the related chapters.

2.5.1 Co-occurrence approaches

Co-occurrence is the simplest approach to identify relationship between two entities that co-occur in the same sentence, abstract, or document [5]. This approach is based on the hypothesis that if two entities are frequently mentioned together, it is likely that they are somehow related. Since two entities might be mentioned together without any relation, most systems use frequency-based scoring schemes to eliminate those relations occurred by chance [8]. The more unlikely the observed relation, the stronger the relation between entities is scored by the system. Co-occurrence approaches tend to achieve high recall but suffer low precision since biomedical texts usually consist of complex sentences which contain multiple entities but only small fraction of these entities are actually related or have relationships. For example, in the AIMed corpus, only 17% of protein pairs that belong to the same sentence actually describe protein-protein interactions [29]. However, the precision of these systems can be improved by applying filtering steps e.g. aggregation of single PPI at corpus level, removal of sentences that do not match lexico-syntactic criteria, or requiring the occurrence of a relation word between two candidate proteins [34]. Another drawback of these methods is that the types of relationships and their direction are not known. Therefore these approaches are not suitable for applications that require fine-grained extracted relations such as annotated PPI pairs for PPI databases.

Since co-occurrence approaches are simple and do not require any linguistic analysis, they are robust in terms of performance time and are still valuable for some applications. For example, co-occurrence approach is often used as baseline method against which other methods are compared when there is no benchmark available [29]. For discovering purposes, e.g. finding relations between genes and diseases, many types of networks are created by applying co-occurrence approaches to a large amount of text, for example, using all of the PubMed abstracts. Such networks do to not provide precise relationships between entities, but they helps organizing the

literature in a way that makes exploration much easier. Examples of recent work based on co-occurrence approaches are [15, 67, 68].

2.5.2 Pattern-based approaches

Pattern-based systems rely on a set of patterns to extract relations. These approaches can be categorized into two groups: manually defined patterns and automatically generated patterns.

Manually defined patterns

Systems based on manually defined patterns require domain experts involve in defining patterns, which is a time-consuming process [36]. These systems differ from each other depending on the level of linguistic analysis employed to define patterns. Earlier systems are based on word forms which usually express patterns using regular expressions, such as *Pro1* * *relation* * *Pro2*, in which *Pro1* and *Pro2* refer to protein names while *relation* refers to the verb which describes the relationship between two proteins [69]. These systems are aimed to extract a limited set of relations given a list of predefined relation words such as *inhibit*, *bind*, *activate*, etc. Obviously, these systems are too simple to achieve satisfactory results [33]. Later systems attempt to define patterns using syntactic analysis of a sentence, such as POS tags, and phrasal structure (e.g. noun phrases, verb phrases, preposition phrases). Such patterns are referred to surface patterns and they do not generalize well on complex sentences [70]. Moreover, the closer examining the text, the more patterns are needed to take account of the large amount of surface grammatical variations in text. To overcome the drawback of surface patterns, some systems incorporate higher level of representation of a sentence i.e. syntactic and semantic structure such as subject-predicate-object or predicate argument structures (PAS) which requires deep analysis of input sentences by full parsing tools [71]. This incorporation leads to some improvements on performance as well as makes the patterns generalize better than those of surface patterns [28].

Overall, manually defined patterns achieve high precision but have relatively poor recall. When such patterns are applied to unseen data, they fail to extract relations that do not occur in the sampling (training) data which used to develop patterns. These approaches are not feasible in practical applications due to their limited generalization therefore they are not adapted well when applying to a new domain.

Automatic patterns generation

To increase the recall of systems based on manually defined patterns, automatically generated patterns are proposed. There are two techniques to generate patterns automatically: by using bootstrapping [72] or generating directly from corpora [73]. In general, bootstrapping techniques do not require a corpus. A small list of relations (e.g. PPI pairs) is given as an input (seeds). The first step is to find patterns that can

extract seeds from text. In the second step, the obtained patterns are then applied to the texts in order to extract new relations of the same type. The initial list of relations is expanded to include newly obtained relations. This process is repeated to find more patterns until no more patterns can be found. To control the way patterns are generated, a distant supervised technique is applied [74, 75].The advantage of the bootstrapping methods is that the initial list of relations is relatively small. However, these approaches are prone to drift and a large number of noisy patterns are generated. In contrast, approaches which generate patterns directly from corpora require a larger number of annotated relations. In both techniques, the linguistic features used to generate patterns are more extensive than those of manually defined, ranging from lexical-based to POS tags, and to syntactic information.

Although the recall of systems using automatic pattern generation can improve, the precision is reduced due to noisy patterns [76, 77]. Moreover, the number of generated patterns is large, some of which are too specific to match unseen text, whereas some are too generic that overmatch any text. Therefore choosing the right patterns to apply for input text is a challenging problem. In order to make these patterns generalize well and to reduce the noisy patterns, many algorithms have been proposed to combine the generated patterns, ranging from dynamic programming to statistical scoring schemes [75, 78, 79].

Overall, automatic patterns generation approaches achieve better performance than those that are manual defined. These systems are capable to generalize well on unseen text. The drawback of these approaches is that the noisy patterns cause the reduction of the precision of these systems. Furthermore, some systems do not take into account the important aspects of extracted relations such as direction and negation of relations.

2.5.3 Rule-based approaches

Rule-based systems rely on a set of rules to extract relations [80, 81]. Similar to the pattern-based approaches, these extraction rules are obtained in two ways: manually defined and automatically generated from training data. In some rule-based systems, the differences between these systems and pattern-based systems are minor since they also use patterns or templates to express rules. Such rule-based systems extend the patterns by adding more constraints to resolve issues that are not easy to express using patterns such as checking negation of relations and determining direction of relations [77, 82, 83]. In contrast, for some rule-based systems, the differences between rule-based and pattern-based systems are clear. Instead of using regular expressions to represent the constraints, these systems rely on a set of rules, which usually express in form of a set of procedures or heuristic algorithms [48, 55, 84]. These rules are then applied to the syntactic structure of a sentence such as a dependency parse tree to extract relations.

Compared to pattern-based approaches, in general, rule-based approaches are more flexible since they are built based on rather abstract levels such as syntactic structures and use grammatical relations and semantic relations of the sentence. Therefore rule-based approaches may generalize well when applied to new domains. Furthermore, the number of rules is relatively smaller than those of pattern-based systems. However, these approaches also suffer from low recall since the defined rules can only cover obvious cases. To improve the recall of these systems, a trade-off between precision and recall can be achieved by relaxing the constraints or by learning rules automatically from training data.

2.5.4 Machine learning-based approaches

With the availability of annotated corpora on biomedical domain, approaches to relation extraction based on machine learning (ML) techniques become ubiquitous [85–90]. Most approaches use supervised learning, in which relations extraction tasks are modeled as classification problems. To build the classification models, data from annotated corpora are transformed into more structural forms by using various NLP tools. Although many ML-based methods have been proposed such as Bayesian network and maximum entropy methods, SVM methods are the most popular one. In general, the ML-based approaches can be categorized into feature-based methods and kernel-based methods.

Feature-based approaches

Feature-based systems use standard kernels, in which each data instance is represented as a feature vector $X=\{x_1, x_2, \ldots, x_n\}$ in an n-dimensional space. These systems treat the SVM classifier as a 'black-box' and mainly focus on defining features that potentially best represent the data characteristics. Various feature types have been proposed to use with the SVM classifiers ranging from lexical to syntactic information [63, 90–95]. The features which are specific to binary relations between two entities (e.g. protein-protein or gene-protein) in the following way:

Bag-of-words (BOW) features: This feature set consists of words that appear after, before, and between two entities. Some systems also use lemma forms of these words and their frequencies as features.

POS features: This feature set consists of the POS tags of words that appear after, before and between two entities.

Shortest path features: This feature set consists of syntactic information derived from the shortest path between two entities which are represented by two nodes of a parse tree (dependency parse tree or phrase structure parse tree).

Graph features: This feature set consists of the labels and weights of two graphs: parse structure sub-graph and a linear order sub-graph connecting two entities. The labels and weights of these two graphs are differentiated.

Beside these common features, many features have been proposed, including distance features (i.e. count the distance between two entities) and cues words (e.g. binding, interaction, activation, etc.) that describe the relations between entities.

In order to improve the performance of the classifiers, most approaches use a combination of these features. Some systems even make use of all of the features with the hope that these features can complement each other. However, when all feature types are used, the number of features increases significantly to hundreds of thousands of features, which in turn, degrades the performance of the system. Therefore, feature selection methods are applied to select the most discriminative features to use with the classifiers [96].

Kernel-based approaches

The limitation of the feature-based approaches is that it cannot make use of the structural representations of instances e.g. syntactic parse trees and dependency graphs. Therefore various kernels have been proposed to tackle this problem [33, 50, 51, 58, 97–99]. The main idea of the kernel methods is to quantify the similarity between two instances by computing the similarities of their representations. Some commonly used kernels are as follows:

Bag-of-words (BOW) kernel: This kernel uses feature vector as unordered sets of words, and calculates the similarity between two compared vectors [53].

Shallow linguistic (SL) kernel: This kernel is defined as the sum of two kernels, the global kernel and local context kernel. The global kernel uses feature set of three BOW vectors (after, before, and between two entities) and count common words in three vectors obtained from two compared sentences. The local context kernel uses surface and linguistic features generated from words on the left and right of two entities [87].

Sub tree (ST) kernel: This kernel uses the syntactic tree representations of sentences. It calculates the similarity between two input trees by counting the number of common sub-trees [51, 98].

Graph kernel: This kernel calculates the similarity between two input graphs by counting weighted shared paths of all possible paths. These paths are obtained from the dependency parse tree and linear order graph [58].

Combination of kernels: As each kernel is design to work with a specific type of input i.e. structural representations of parse trees or sequential words of sentences and it may have different strength and weakness. Therefore the combination of kernels has also been used in some approaches to relation extraction [50, 99]. This combination shows a slight increase in performance, however, the computational resource needed to train the classifiers also increases significantly [100].

2.5.5 Performance comparison of existing approaches to relation extraction

A meaningful performance comparison of existing relation extraction approaches can serve as important instruments both for perspective researchers who are interested in the relation extraction methods and for end users who want to apply automatic relation extraction techniques. However, such comparisons are not always available due to many reasons. First, these approaches target various relation types such as genes-diseases, drug-drug interactions, and protein-protein interactions. Second, many relation extraction systems evaluate their performance on different datasets which have various characteristics such as annotation policy, the ratio of positive and negative examples. Third, even if these systems are aimed to extract the same type of relation and are evaluated on the same test set, their results are not comparable due to different evaluation settings e.g. training data are split using instance-based vs. document-based level. Therefore it is not straightforward to directly compare the performance of existing systems in a fair and meaningful manner.

To facilitate a fair and straightforward performance comparison between extraction approaches, many competitive evaluations have been held. The official results from these competitions show that performances of the participant systems are far from achieving satisfaction results in order to directly integrate into real life applications. For examples, the official results of the BioCreative II.5 community challenge on extracting protein interactions show that the best system performance achieves an F-scores only above 30% [66], whereas recent evaluation of the BioNLP'11 Shared task for the GENIA event task show that the results in terms of F-scores range from 12% to 50% using strict matching criteria [60]. Furthermore, there have been some attempts to independently evaluate the existing PPI extraction methods to obtain a realistic performance on biomedical text. A study by [34] shows that performance of a typical rule-based system on the PPI extraction tasks is better than the state-of-the-art ML-based systems when evaluated using cross-corpora and cross-learning criteria. This finding is once again confirmed in a recent study by [33] on comprehensive benchmark of kernel methods to extract PPIs. In this study the authors conclude that the performance of kernel-based methods is very sensitive to parameter settings and depends largely on the corpora of which they are trained on and evaluated on. This means that extrapolating their measure performance on arbitrary text is highly problematic. Table 2.4 and Table 2.5 present performance of existing methods for PPI and biological events extraction tasks, respectively. The results show that ML-based approaches perform slightly better than the other approaches. However, it is unclear which method is the best to perform arbitrary relation extraction task. For example, the kernel-based system proposed by [45] performs better than the pattern-based system of [74] on the AIMed corpus but the pattern-based system of [74] outperforms system of [45] on the BioCreative-PPI corpus.

Table 2.4. Performance comparison of existing PPI approaches on AIMed and BioCreative-PPI (BC-PPI) corpora. Systems run on the BC-PPI corpus are evaluated at document level, whereas systems run on the AIMed corpus are evaluated at mention level.

Approach	Recall	Precision	F-score	Corpus	Ref.
Co-occurrence	51.5	22.8	31.6	AIMed	[34]
Pattern-based	71.8	48.4	57.3	AIMed	[74]
Rule-based	50.0	40.0	44.0	AIMed	[29]
Feature-based	71.9	60.0	65.2	AIMed	[63]
Kernel-based	66.6	62.7	64.2	AIMed	[45]
Kernel-based	34.5	53.1	37.4	BC-PPI	[45]
Pattern-based	43.4	43.4	42.9	BC-PPI	[74]

Table 2.5. Evaluation results of some participant teams of the BioNLP'11 Shared Task on Genia event extraction using strict matching criteria.

Team	Approach	Recall	Precision	F-Score	Ref.
UMASS	ML-based	43.89	57.96	49.95	[101]
UTurku	ML-based	45.59	53.00	49.02	[102]
MSR-NLP	ML-based	44.63	50.18	49.02	[103]
ConcordU	Rule-based	39.06	53.44	45.13	[64]
CCP-BTMG	Pattern-based	29.55	55.13	38.48	[73]

2.6 Conclusion

In this chapter we have described the components of a typical relation extraction system such as text preprocessing, named entity recognition (NER), parsing, and relation extraction. We explained the role of the text preprocessing step, the challenges of the NER task in biomedical texts and how it affects the performance of a relation extraction system. We have discussed the natural language processing (NLP) steps used to convert unstructured text into structured representations and explained how these can be used for relation extraction tasks. Furthermore, we

discussed the application of machine learning (ML) methods with a focus on support vector machines for the relation extraction tasks and the biomedical corpora required by these methods. Finally, we briefly presented four main approaches to the relation extraction tasks, namely co-occurrence, pattern-based, rule-based, and ML-based approaches and discussed their performances on typical tasks. The details for specific relation extraction task are being discussed in subsequent chapters.