Relation Extraction of Transcription Factor to Genes from Biomedical Literature

THESIS

Submitted in partial fulfillment of the requirements of BITS F423T / BITS F421T, Thesis

by

Author:	Ashish Baghudana
ID No:	2011B1A7575G
Supervisor:	Prof. Dr. Burkhard Rost
Advisor:	Juan Miguel Cejuela
Date:	December 8, 2015

BITS-Pilani, K. K. Birla Goa Campus

CERTIFICATE

This is to certify that the Thesis entitled, Relation Extraction of Transcription Factor to Genes from Biomedical Literature submitted by Ashish Baghudana, ID No. 2011B1A7575G in partial fulfillment of the requirements of **BITS F 423T / BITS F421T** Thesis embodies the work done by him / her under my supervision.

Prof. Dr. Burkhard Rost
Alexander von Humboldt Professor
Technische Universität München

Date: 15th December, 2015

Acknowledgments

First, I wish to thank Prof. Dr. Burkhard Rost for giving me the opportunity to work in his group. Rostlab was an excellent fit for me, given my interest in Computer Science and Biological Sciences.

A very special thanks to my advisor and mentor, Juan Miguel Cejuela, for his patience, guidance and support throughout the thesis as well as before starting when he patiently answered all my queries. The weekly meetings, code reviews, feedback and suggestions were extremely valuable during the entire thesis. I extend my thanks to Tim Karl helping me out with the Rostlab account as well as other technical difficulties I faced during my code implementation. I want to thank the entire Rostlab for welcoming a foreign student and creating a very good learning atmosphere. I also wish to thank my on=campus mentor Dr. Sumit Biswas for his support.

Last but never the least, I want to thank my family for their unconditional love, support and encouragement. Without them, this thesis would not have been possible.

Abstract

This thesis introduces a novel relation extraction method for deriving the relations of transcription factors and proteins from the biomedical literature. To the best of my knowledge, my method is the first ever published to be specifically built for this task. To train the method, a new corpus *relna*, consisting of 150 MEDLINE abstracts was annotated. *relna* was semi-automatically constructed, using GNormPlus. It was found out that over 90% of the transcription factor-protein relations were present in the same sentence.

We therefore, focused on extracting relations from a single sentence. My model uses support vector machines and makes heavy use of dependency graph-based features. On *relna* corpus, my method achieved a precision of 69.1%, recall of 69.9% and F-score of 69.3%. The Relation Extraction module is built on the *nalaf* framework. This provides a generic framework for any named entity recognition and relation extraction task on biomedical literature, and allows for easy extendibility.

The corpus is publicly available in PubAnnotation format at `http://pubannotation.org/projects/relna` and the method is available on Rostlab's GitHub page `https://github.com/Rostlab/relna`.

Contents

1 Introduction

1.1 Expansion of Biomedical Data

Research in biological sciences has not been more active in the past. Over a million articles are published in life sciences and biomedical information every year and indexed by MEDLINE, an online bibliographic database compiled by United States National Library of Medicine (NLM). As of 2015, MEDLINE indexes over 24 million articles, with a temporal coverage from 1951. The rate of growth of MEDLINE is show in Figure 1.1.

1.1.1 The Need for Text Mining

Given the time span and ever-inreasing number of articles that are published, it is not an easy task for researchers to keep up with all developments in their field. Moreover, it is also difficult to summarize or aggregate findings for new researchers in the area. Given the textual nature of scientific literature, text mining techniques help researchers extract structured information from this large dataset, without manually reading through each article. Over the years, biological text mining research has focusing on the identification of biological “entities" in text, such as protein names, species, subcellular locations etc. and the extraction of complex relationships between these entities.

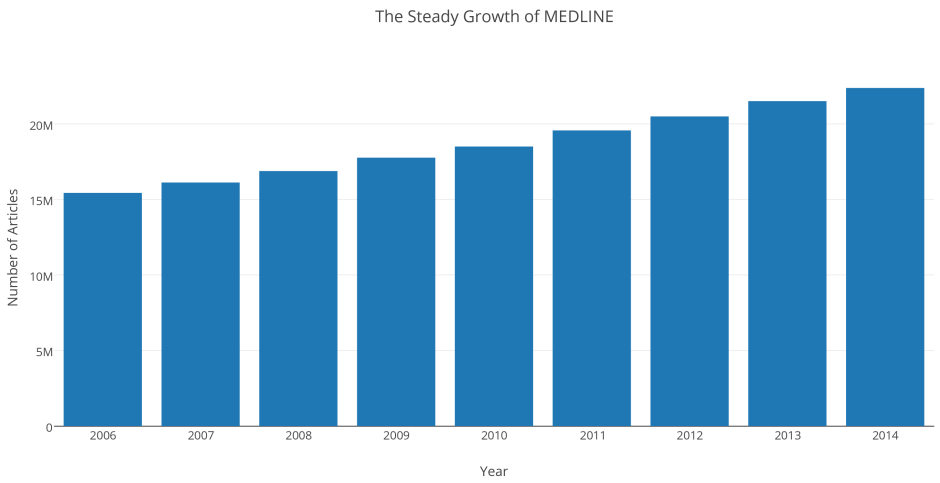

Figure 1.1: MEDLINE has been growing almost linearly with the addition of close to 1M articles annually.

1.2 Named Entity Recognition

Named Entity Recognition (NER) is one of the key components of Biomedical Natural Language Processing (BioNLP). In any BioNLP pipeline, the first step involves recognition of these entities. While named entity recognition in other domains has reached incredible accuracy (of almost 94%) [12][3], biomedical NER still remains a challenge. This is predominantly because of the variance in biomedical literature. Several corpora have been published for supervised named entity recognition, including the GENIA corpus [7], NLPBA corpus[8], GeneTag [19] and BioCreative [22]. One of the best performing systems is GNormPlus [21], achieving an F-measure of 78.8% on single gene types and 86.7% on gene, gene families and protein domains.

Gene Mention Type Schema	Precision	Recall	F-Measure
Gene/Family/Domain	87.1%	86.4%	86.7%
Single Gene Type Only	78.4%	79.2%	78.8%

Table 1.1: Performance of GNormPlus across Gene/Family/Domain and Single Gene only. Reproduced from [21]

1.3 Relation Extraction

Biomedical Relation Extraction forms the next step in the BioNLP pipeline. Over the last decade, several biomedical relation extraction tasks have been performed. Protein-Protein Interactions[17], Drug-Drug Interactions[18], Protein-Organism-Subcellular Location Relations[10] and Disease-Drug-Adverse Reactions Text Mining[20] are just some examples of diverse relation extraction tasks. The major aim of Biomedical Relation Extraction is to uncover new and existing relations from biomedical literature with high accuracy and efficiency, and store this in structured formats for easy analysis. The past decade has garnered significant interest in binary, as well as complex relations among biomedical entities. These relations could be potentially used for therapeutic approaches and drug discovery [24].

1.3.1 Relation Extraction as a task in Binary Classification

Relation Extraction is a complicated NLP task and is usually not performed directly. It often involves a method of transformation such that it is suitable for a machine learning task. In this thesis, we convert the problem of relation extraction into a task of binary classification.

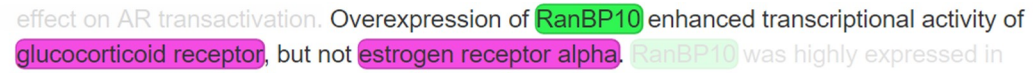

Figure 1.2: The figure shows a a sample text that has been annotated for relations in tagtog [4].

As with any machine learning method, this involves several steps, including forming the dataset as a list of tuples with a class name (dataset creation), data preprocessing, feature generation, feature selection, learning a model and finally classification. Sample text annotated using tagtog [4] is converted into features by extensive feature generation, the structure of which is shown in Table 1.2.

Entity 1	Entity 2	Feature 1	...	Feature n	Class
RanBP10	Glucocorticoid Receptor	1	...	0	True
RanBP10	Estrogen Receptor Alpha	0	...	1	False

Table 1.2: Converting Relation Extraction to Binary Classification. Feature Generators extract semantic features from text and convert them to (typically) binary values and label them as True or False relations.

1.4 Problem Motivation: Transcription Factors Relations

Transcription factors are a major class of proteins that affect almost all life processes. They directly influence gene regulation and expression. Their disruption is a significant cause for several types of cancer. Indeed, their interactions could hold the key to controlling cancer progression, amongst several other diseases. We therefore, focused

on the relations between transcription factors and their associated genes and gene products. We examined the genes that transcription factors *transcribe* and the gene products that *modify* the behavior of these transcription factors. We feel these relations are biologically very relevant and can be used to build large scale maps of transcription factor interactions.

2 Corpus Construction

relna is a corpus of interactions between transcription factors and gene or gene products (GGP). Transcriptional regulation is one of the fundamental ways in which gene regulation occurs across several organisms. *relna* seeks to annotate functional relationships between

- transcription factors and the genes they transcribe (transcription factor *transcribes* gene)
- transcription factors and proteins that modify the behavior of these transcription factors (protein *modifies* transcription factor)

to make identification of such interactions possible from natural text.

A growing amount of research attributes dysfunctional transcription factors to several genetic irregularities and diseases (several oncogenes are transcription factors [9]). Additionally, several tumor suppressor genes are also transcription factors. Understanding their interactions with different genes and proteins could help design better cures.

relna consists of 140 abstracts semi-automatically annotated for transcription factors (TF) and gene or gene products (GGP). Both the TF and GGPs are normalized to both their Entrez Gene ID and UniProt ID. The relations between these transcription

factors and GGPs were manually annotated using tagtog [4].

2.1 Document Extraction

1. SwissProt was filtered for proteins with the following two criteria:
   - They belong to *Homo sapiens*
   - They are annotated with the Gene Ontology Term *GO:0003700* [1] - Transcription Factor Activity, Sequence-Specific DNA Binding
2. This gave a list of over 1100 proteins
3. For each of these proteins, a list of publications that have cited this specific protein are obtained
4. From among the list of publications, we further filter those papers that been cited for **INTERACTION WITH**
5. This forms the *base corpus* of 1461 abstracts with no entity mentions of relations

2.2 Entity Recognition and Tagging

A random sample of 140 abstracts is picked from the *base corpus* for named entity recognition. We use state of the art GNormPlus [21] for tagging our abstracts with gene or gene product names. GNormPlus currently annotates genes, gene families, species and domain-motifs. GNormPlus applies its taggers to the entire PubMed and stores their results in PubTator with nightly updates. GNormPlus was accessed programmatically through its RESTful API [14] and returns all results in the PubTator

format [15]. GNormPlus also performs gene normalization against the Entrez Gene Database [13]. The information of gene families, species and domain-motifs is ignored and *relna* focuses solely on genes (or gene products).

```
<PMID>|t|<TITLE>
<PMID>|a|<ABSTRACT>
<PMID> <START OFFSET 1> <LAST OFFSET 1> <MENTION 1> <TYPE 1> <IDENTIFIER 1>
<PMID> <START OFFSET 2> <LAST OFFSET 2> <MENTION 2> <TYPE 2> <IDENTIFIER 2>

<PMID>|t|<TITLE>
<PMID>|a|<ABSTRACT>
<PMID> <START OFFSET 1> <LAST OFFSET 1> <MENTION 1> <TYPE 1> <IDENTIFIER 1>
<PMID> <START OFFSET 2> <LAST OFFSET 2> <MENTION 2> <TYPE 2> <IDENTIFIER 2>
```

Listing 2.1: PubTator Format as Tab Separated Values (TSV) having 6 columns

2.3 Entity Normalization

Each gene or gene product annotated by GNormPlus is cross-referenced with SwissProt using the Entrez Gene ID. UniProt has its own mapping service that allows a crude one-to-many mapping between Entrez Gene ID and UniProt IDs. I devised a priority selection for one-to-one mapping between Entrez Gene ID and UniProt ID by giving a higher preference to SwissProt. Assuming a corresponding SwissProt ID is not found, then TrEMBL IDs are obtained and sorted according to score based on annotation quality (already given in UniProt). The TrEMBL ID with the highest score is chosen as the corresponding UniProt ID. This is an improvement over a naive method of

storing all corresponding UniProt IDs for the particular Entrez Gene ID. However, it is important to note here that gene name normalization is not fully accurate and can be slightly error prone for two reasons:

1. Firstly, gene normalization is not entirely accurate - for instance, normalization sometimes loses information about species and we have realized that some Gene IDs are from *Mus musculus*. In a very small number of cases, the normalized protein may not match the original.

2. Secondly, Swissprot mappings themselves need not be one-to-one. While in most cases, we see only a single SwissProt ID for a Entrez Gene ID, it is indeed possible for multiple SwissProt IDs (in case of isoforms of a protein). In this case, we naively choose the top scored result as our equivalent mapping.

After obtaining the UniProt IDs, we check each protein for the presence of the GO Term *GO:0003700*, which indicates transcription factor activity. All gene or gene products having this term are labeled transcription factors, and those that do not are left as GGPs.

In the final step, PubTator files are converted to the tagtog [4] format (*.html and .ann.json*) using a Python library PubTator2Anndoc. The tagtog format can be accessed at `https://github.com/tagtog/tagtog-doc/wiki/tagtog-document-formats`.

2.4 Relation Annotation

The converted files are uploaded on to tagtog. I manually annotated relations between transcription factors and their associated proteins using a set of annotation guidelines that are mentioned below.

2.4.1 Annotation Guidelines

The following annotation guidelines are used to tag entities and relations. As is described, even though entity tagging was performed by GNormPlus, some corrections were made to include abbreviations.

Entities

1. **Entity Types**: We annotate entities of the type Transcription Factor and Transcription Factor-binding.

2. **Normalization**: An entity is normalizable if it can be looked up in a standard database. Entities must be searchable in NCBI Gene Database and UniProt using Gene ID and UniProt ID.

3. **Entity Boundaries**: Often GNormPlus misrepresents boundaries. For example, an entity may be incorrectly tagged as `Estrogen receptor (ER`. This is corrected to `Estrogen receptor` and `ER` while reviewing tags

4. **Rule of Acronyms**: Wherever the full name is annotated, also annotate the abbreviation / acronym. For instance, in `Androgen receptor (AR)`, `Androgen receptor` and `AR` are both annotated

5. **Unidentified Entities**: If an entity is recognized at one place, and not at the other, annotate all mentions of the entity.

Relations

There are two broad categories of relationships.

1. **Modify**: Transcription Factor-binding proteins that regulate, modify or co-activate other transcription factors

2. **Transcribe**: Genes that are transcribed by the transcription factor

As a rule of thumb, I have not annotated any relations that do not imply physical binding.

2.4.2 Meaningful Relations

A relation is meaningful if either of following two conditions is true:

1. We can find the written words used by the author that implies that there can be a relation (in this case, the relation is not indirectly implied, but is clearly written)

    - **Example**: In this work, we investigate the phosphorylation of the N-terminal heterodimerization (PAS) domain of `HIF-1alpha` and identify Ser247 as a major site of in vitro modification by `casein kinase 1delta` (`CK1delta`). Annotate the relation between HIF-1alpha and casein kinase 1delta as the word *modify* clearly indicates a physical relation.

2. We can *infer* the relation from the context and it can be verified from UniProt

    - Example: We further showed that `CCAR1` is required for recruitment of `AR`. While the relation is not explicit, the sentence indicates that CCAR1 *possibly* interacts with AR. In these situations, the UniProt entry for AR (Androgen Receptor) was checked to verify the relation between the two entities. Only then was the relation annotated.

2.5 Corpus Statistics

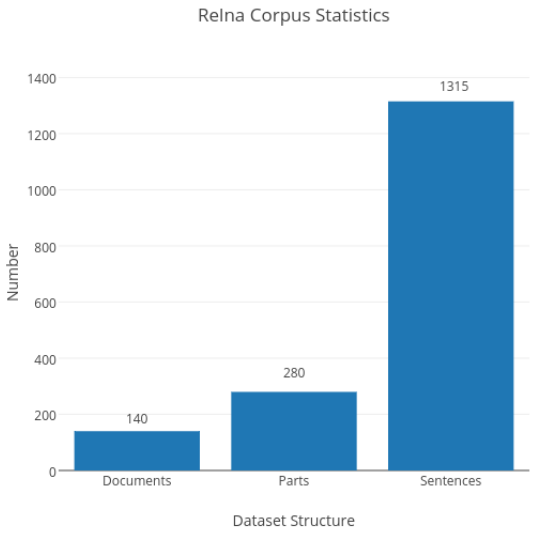

Figure 2.1: The number of documents, parts and sentences in *relna*

relna consists of 140 documents, covering over 1300 sentences. In these documents, there are totally 865 unique entities - 446 genes or gene products and 416 transcription factors. There are close to 2800 mentions of entities.

Every document has an average of 3.186 GGP mentions and 2.971 transcription factor mentions. The entities in the documents repeat significantly as the average number non-unique of GGP mentions is at 10.178 and average number of non-unique transcription factor mentions is at 9.614. Furthermore, *relna* has an average of 1.75 unique relations per document.

These statistics are visually presented below.

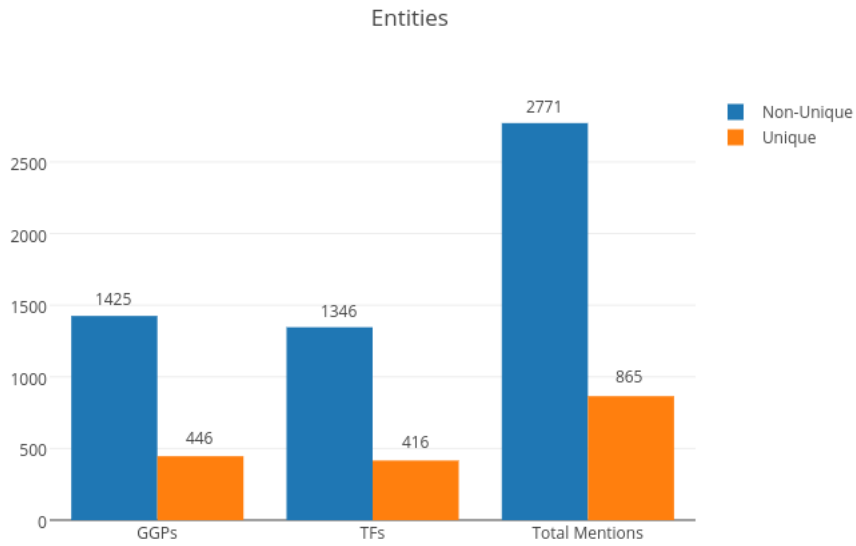

Figure 2.2: The number of unique and non-unique entities in *relna*

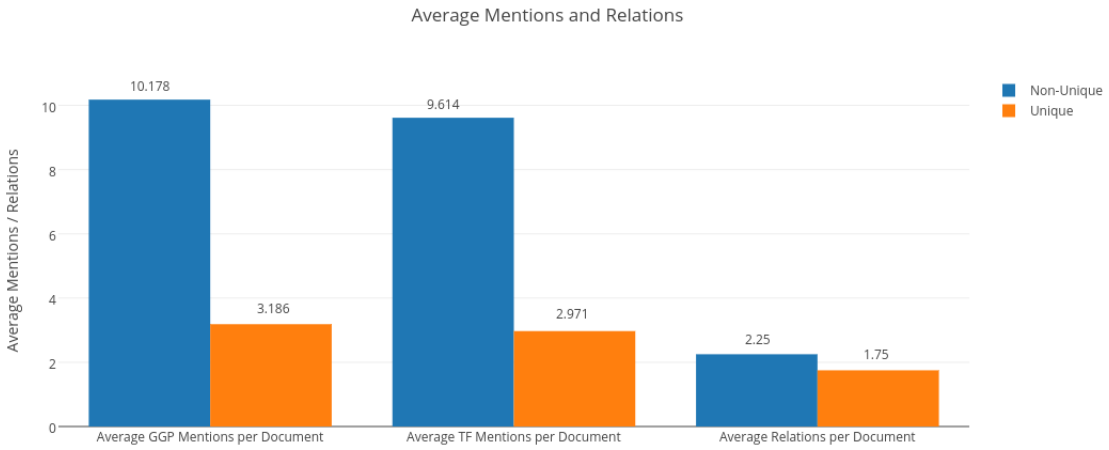

Figure 2.3: The number mentions and relations in *relna*

3 Background Theory

My thesis uses significant background theory from NLP and Machine Learning, specifically Support Vector Machines. Our features derive heavily from Dependency Parsing and Constituency Parsing, both of which are described below. This chapter concludes with a discussion on Support Vector Machines.

3.1 Parsing in Natural Language Processing

Parsing in NLP is defined as the process of assigning structural descriptions to sequences of words. The structural descriptions follow from a rule set, often called a grammar. A parser accepts as input a sequence of words a description of possible structural relations that may hold between words or sequences of words. Predominantly, we use two kinds of parsing in NLP - constituency parsing and dependency parsing.

3.1.1 Constituency Parsing

A constituency parse tree breaks a text into sub-phrases and organizes these sub-phrases in a hierarchical fashion. Non-terminals in the tree are types of phrases, the terminals are the words in the sentence, and the edges are unlabeled. For an example sentence of

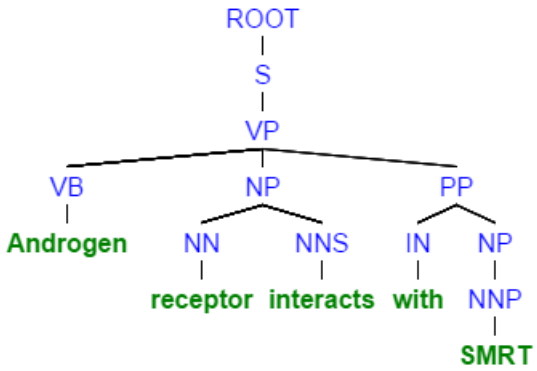

Figure 3.1: The Constituency Parse Tree for "The interaction between Androgen Receptor and SMRT"

"The interaction between Androgen Receptor and SMRT", the constituent parse tree is shown in Fig 3.1.

Here, "The interaction" and "Androgen Receptor and SMRT" are noun phrases that are linked together by a preposition, "between". Machine Learning approaches are used to make Constituency Parsing as accurate as possible.

3.1.2 Dependency Parsing

A dependency parser analyzes the grammatical structure of a sentence, establishing relationships between "head" words and words which modify those heads. As you can see from Fig 3.2, the root words for both sentences indicate "interaction", even though the sentence structure varies.

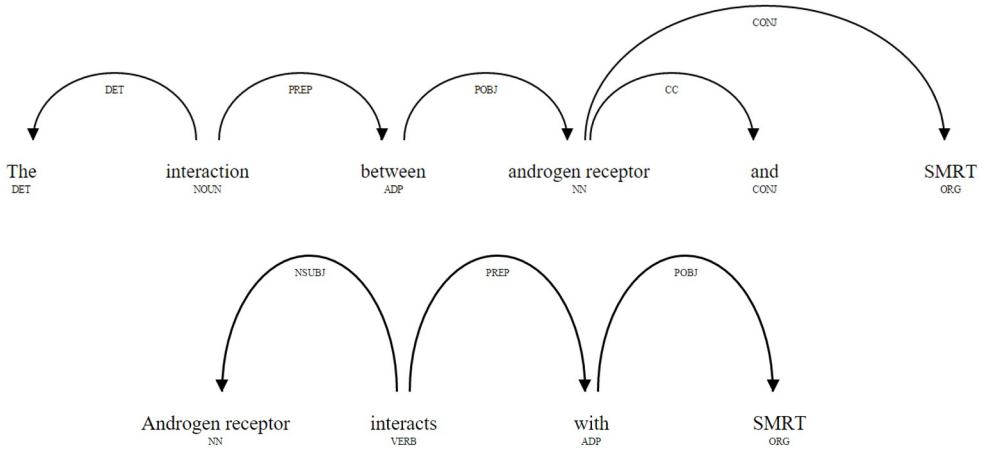

Figure 3.2: The Dependency Parse Tree for "Androgen Receptor interacts with SMRT" and "The interaction between Androgen Receptor and SMRT"

3.2 Support Vector Machines

Machine Learning is a method of teaching computers to make and improve predictions or behaviors based on some data. Machine Learning is a another phrase for pattern recognition - the act of teaching a program to react to or recognize patterns. There are several types of machine learning algorithms that can learn these patterns from data. Machine learning tasks can be broadly classified into supervised learning and unsupervised learning. In supervised learning, the algorithms are given a set of data along with their labels and the goal for algorithms is to learn a function that maps the input data to their correct given labels. In unsupervised learning, the goal is the same, however, the training data is given without assigned labels and so the algorithms must assign the correct implicit labels.

Support vector machines (SVM) are supervised learning models that can be used for the task of classification and regression. SVMs are widely used for the task

of classification, that is, classifying the data into predefined classes/labels. Support vector machines are very effective in high-dimensional space, that is, when instances have many features. They are also extremely memory efficient because they use only a subset of training points in the decision function (called support vectors). SVMs employ different kernels to classify data. Kernels are nothing but similarity functions. The default kernel in any SVM is the linear kernel.

Given n labeled examples $(x_1, y_1), \ldots, (x_n, y_n)$ with labels $y_i \in \{1, -1\}$, we want to find the hyperplane $< w, x > +b = 0$ (i.e. with parameters (w, b)) satisfying the following three conditions:

1. The scale of (w, b) is fixed so that the plane is in canonical position w.r.t. $\{x_1, \ldots, x_n\}$. i.e.,
$$\min_{i \leq n} | < w, x_i > +b| = 1$$

2. The plane with parameters (w, b) separates the $+1$'s from the -1's. i.e.,
$$y_i(< w, x_i > +b) \geq 0 \text{ for all } i \leq n$$

3. The plane has maximum margin $\rho = 1/|w|$. i.e., minimum $|w|^2$.

Of course there may not be a separating plane for the observed data. Let us assume, for the time being, that the data is in fact linearly separable and we'll take care of the general (more realistic) case later.

Clearly 1 and 2 combine into just one condition:

$$y_i(< w, x_i > +b) \geq 1 \text{ for all } i \leq n.$$

Thus, we want to solve the following optimization problem,

$$\text{minimize } \frac{1}{2}|w|^2$$

over all $w \in R^d$ and $b \in R$ subject to,

$$y_i(< w, x_i > +b) - 1 \geq 0 \text{ for all } i \leq n.$$

This is a very simple quadratic programming problem. There are readily available algorithms of complexity $O(n^3)$ that can be used for solving this problem. For example the so called interior point algorithms that are variations of the Karmarkar algorithm for linear programming will do. But, when n and d are large (tens of thousands) even the best QP methods will fail. A very desirable characteristics of SVMs is that most of the data ends up being irrelevant. The relevant data are only the points that end up exactly on the margin of the optimal classifier and these are often a very small fraction of n.

4 Relation Extraction Framework

The relation extraction framework for *relna* is based on the *nalaf* framework and uses some of its classes and data structures. The aim of the relation extraction framework is to to have tight integration with *nalaf*, and have an extensible interface, so that anyone can easily customize the pipeline for their needs.

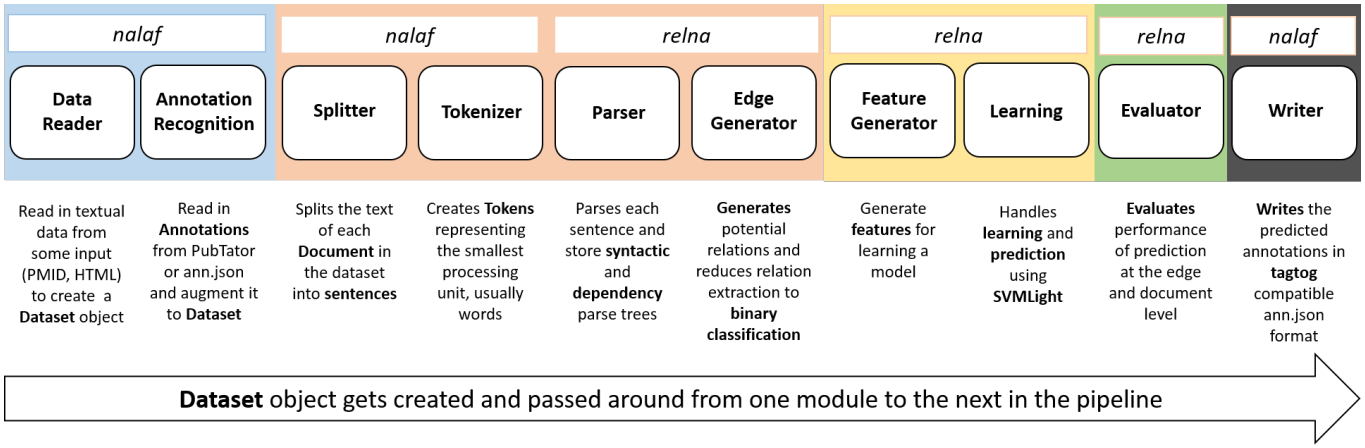

Figure 4.1: The relation extraction framework of *relna*

4.1 Data Preprocessing

Data is read in through the `HTMLReader` and `AnnJsonAnnotationReader` interface and stored in a `Dataset` object. Through the pipeline, this `Dataset` object is passed around for processing, feature generation, feature selection, training and evaluation of performance.
Data Preprocessing for relation extraction involves four major steps as depicted in Fig 4.1. These are:

1. **Splitting**: Each `Document` in the `Dataset` is split into sentences.
2. **Tokenizing**: Each sentence in the `Document` is split into its constituent tokens, the smallest processing unit for the pipeline.
3. **Parsing**: Each sentence is parsed, and its corresponding tree structure is stored.
4. **Edge Generation**: A list of potential relations are generated, so they can be classified.

4.1.1 Parsing

Parsing is the most time consuming step of preprocessing where each sentence is analyzed for their phrase structures and dependencies. As described in the previous chapter, constituency parsing establishes the hierarchy within the sentence and groups sub-structures of the sentences. Dependency parsing on the other hand establishes relations between two words in a sentence. Our model uses features derived from constituency and dependency parse trees extensively for relation classification.

4.1.2 GENIA+PubMed Dataset

The GENIA corpus is the primary collection of biomedical literature compiled and annotated within the scope of the GENIA project. The corpus was created to support the development and evaluation of information extraction and text mining systems for the domain of molecular biology. The corpus contains 1,999 Medline abstracts, selected using a PubMed query for the three MeSH terms “human", “blood cells", and “transcription factors". The corpus has been annotated with various levels of linguistic and semantic information. Part-of-speech and syntactic (phrase structure) annotation has been created for all of the 1999 abstracts of the primary GENIA corpus. The annotation scheme of the GENIA Treebank has been designed based on the Penn Treebank II (PTB) bracketing guidelines [2]. The GENIA Treebank is the most widely applied corpus for training and adapting parsers to biomedical domain texts and has been applied to several parsers including Stanford Parser, BLLIP Parser, Enju Parser and GDep (Genia Dependency) Parser.

4.1.3 BLLIP and SpaCy Parsers

We use the Charniak-Johnson parser, also known as the BLLIP Parser for constituency parsing. The algorithm for BLLIP parser was developed by Charniak and Johnson [5]. BLLIP Parser also performs dependency parsing and parses to Stanford Dependencies using a Python library, PyStanfordDependencies. BLLIP Parser is trained on GENIA+PubMed dataset and tends to be more accurate than spaCy for dependency parsing. spaCy parser, trained on OntoNotes 5, is however significantly faster than BLLIP Parser. When constituency parsing is not required, spaCy parser outperforms BLLIP parser.

4.1.4 Edge Generation

The edge generation phase of relation pipeline generates potential relations in the text. Currently, any two entities from the same sentence form a potential relation. This is called `Simple Edge Generation`. Another approach to edge generation is to filter sentences specifically for certain keywords. Edge generation eventually will also involve multiple sentences.

4.2 Feature Extraction

There are three major sources of features for relation extraction - dependency parse trees, constituency parse trees and sequences of text. A variety of features are extracted from all these sources and combined for optimal performance. An overview of the features can be presented as follows.

4.2.1 Sentence Features

Sentence features are extracted for every potential relation that is a part of the sentence. Sentence features include bag of words (BOW) and stemmed BOW of tokens in the sentence, frequency features like the number of entities in the sentence and n-gram features of up to window size of 3. We also extract specific keywords from the sentence such as "interact" or "bind". This list of keywords is provided based on the domain of relation extraction. We also look at the order of entities in a sentence and the annotation types of all entities in a sentence.

4.2.2 Token Features

Token features are extracted for tokens that are part of the entities and for tokens that are in a linear dependency with the entities. Every entity is represented by a head token, which is calculated by a simple heuristic. If an entity consists of a single token, then that token represents the entity as head token. On the other hand, if an entity consists of multiple tokens, a dependency based score is calculated for all entity tokens and the token with highest score represents the entity as head token.

Token features include token text, masked token text, stem, POS tag, etc. Most of the features are binary and some are integer-valued features. There are features for binary tests on a token, e.g., whether the first letter of token is in capital case or not (capitalization test), whether some letter in the middle is in capital case or not, etc. Other binary tests include presence of a hyphen, presence of forward slash, presence of backward slash, presence of digits, etc. Character bigrams and trigrams of the token also contribute to the token features

4.2.3 Linear Context

Token features are also extracted for tokens that are present in the linear and dependency context. A linear context of length 3 is considered, i.e., features are extracted for the next and previous 3 tokens relative to the token under consideration. A dependency chain of length 3 is considered for the dependency context. Incoming and outgoing dependencies are considered for dependency-related features. For example, in the case of incoming dependencies, features are extracted for the source / from token and its incoming and outgoing dependencies are considered too. This goes on up to a dependency depth of 3. In addition to the token features, the dependency edge types are also considered while extracting dependency chain features.

4.2.4 Dependency Path Features

Many features are extracted depending on the dependency graph. Using the Floyd-Warshall algorithm, the shortest path between any two entities in a potential relation is calculated. For the purpose of extracting the shortest path, an undirected graph of dependencies is considered. Note that an undirected graph is considered only for the purpose of extracting the shortest path. Most of the dependency-based features depend on this shortest path. While extracting the features, the original direction of the edges in the shortest path is also taken into consideration. The length of the shortest path contributes an integer-valued feature in addition to the binary feature for each length. Token features are extracted for terminal features of the shortest path, which are head tokens of the entities. Some of the other path related features include token features of the internal tokens in the path, features for every edge in the path, features for internal edges in the path, keywords in path, lowest common ancestor and root of the dependency parse tree.

4.2.5 N-gram Dependency Features

All entities are represented by their respective head tokens. The shortest path between two entities is actually a shortest path between the head tokens. However, there need not be a single shortest path between two entities. There can be multiple paths between two entities with same distance or same minimum distances. All such paths with minimum distance are computed and features are extracted for each one of them.

For every such minimum distance path from the set of minimum distance paths, parts of the paths are considered for N-gram features. For example, a set of all 2 consecutive tokens are considered for 2-gram features and a set of all 3 consecutive tokens are considered for 3-gram features, etc.. 2-, 3- and 4-gram features are extracted

from all such paths. These features also include token features that are part of the corresponding set, dependencies in the set, directions of dependencies in the set, etc.

4.3 Training, Cross Validation and Classification

We used SVMLight and SVMLight TK for classification of relations and used cross validation techniques for evaluation of results.

4.3.1 Cross Validation

A 5-fold cross validation is used where every fold consists of 20% of the documents. We use two strategies for evaluation - first, an 80:20 split, and second, an 60:20:20 split. In the 80:20 split, 80% of the total data is used for training, and the remain 20% is used for testing. In the 60:20:20 split, 60% of the total data is used for training, 20% for development and 20% for test. The training data is used to train the model using SVMLight uniformly. When there is a development set, we use to select features and tune the hyperparameter (mainly, the tradeoff between the margin of the hyperplane and the misclassification error). This is demonstrated in Fig. 4.2.

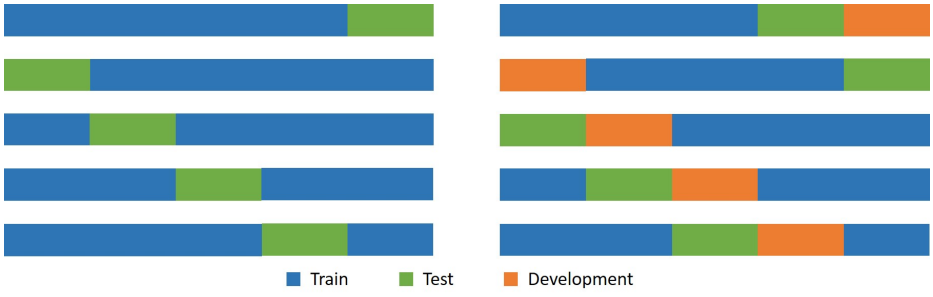

Figure 4.2: The two methods of cross validation, 80:20 split and 60:20:20 split

4.3.2 SVM Classification

I used SVMLight [6] for classification of relations. The data representation used by SVMLight is given in Listing 4.1

```
<line> .=. <target> <feature>:<value>...<feature>:<value> # <info>
<target> .=. +1 | -1 | 0 | <float>
<feature> .=. <integer> | "qid"
<value> .=. <float>
<info> .=. <string>
```

Listing 4.1: Each line in the SVMLight format is an instance labeled with its class, and the set of features

We also used SVMLight TK [11] whenever we used constituency parse trees along with other features for parsing as shown in Listing 4.2.

```
<line> ::= <target> <set-of-vectors> | <target> <trees-and-vectors>
<set-of-vectors> ::= <vector> | <vector> <begin-vector> ... <vector> <end-vector>
<set-of-trees> ::= <begin-tree> <tree> ... <begin-tree> <tree> <end-tree> | " "
<trees-and-vectors>::= <set-of-trees><blank><set-of-vectors>

<vector> ::= <feature>:<value> ... <feature>:<value>
<target> ::= +1 | -1 | 0 | <float>
<feature> ::= <integer> | "qid"
<value> ::= <float>
```

```
<begin-tree> ::="|BT|"
<end-tree> ::="|ET|"
<begin-vector>::="|BV|"
<end-vector> ::="|EV|"

<tree> ::= <full-tree> | " "
<full-tree> ::= (<root> <full-tree> .. <full-tree>) | (<root> <leaf>)
<leaf> ::= <string>
<root> ::= <string>
```

Listing 4.2: Each line in the SVMLight format is an instance labeled with its class, and the set of features

4.3.3 Evaluation Modes

We used two modes of evaluation - edge level and document level performance. They are explained briefly below.

Unique

In the unique evaluation mode, a set of unique relations of each type is made for every document in the corpus. Two relations are considered unique if the text of the two entities and the relation type matches. A predicted positive relation is said to match with the original unique relation if and only if the predicted relation for the document is found in the set of unique relations for that document. If the predicted relation is not found in the set, it is either classified as a false positive if it is a positive prediction

or a true negative if it is a negative prediction. We believe that for a biologist, unique relations matter much more than non-unique relations.

Non-unique

Simply put, the results of non-unique evaluation mode is the direct output of SVM classification. The predictions are compared with original relations and the results are computed. For a particular document in the corpus, it may happen that a relation is repeated several times across the document. In this evaluation mode, such repetitions are considered separately and therefore, this evaluation mode is called non-unique evaluation mode. For this mode of evaluation, a predicted relation is said to match with an original relation if and only if the text offsets as well as the text of both entities match.

5 Results and Discussion

Support Vector Machines come in different flavors, each of which can be tweaked to a large extent. Our evaluations for performance measures on the SVM based machine learning method report precision, accuracy and F-measure. Precision is a measure of exactness, that denotes what percentage of the tuples identified as positive are indeed correct, whereas Recall is a measure of completeness, that denotes what percentage of the positive tuples have been identified correct. F-measure is merely the harmonic mean of Precision and Recall. These three measures are fundamentally based on the number of true positive (tp), false positive (fp) and false negative (fn) predictions and can be mathematically represented as follows.

$$Precision = \frac{tp}{tp + fp}$$

$$Recall = \frac{tp}{tp + fn}$$

$$F\text{-}measure = 2 \times \frac{precision + recall}{precision \times recall}$$

We performed extensive evaluation of relation extraction on the relna corpus, using different variations in the SVM method. The evaluation strategy used is a 5-fold

cross validation as described previously. The results are presented below, showing precision, recall and f-measure for each fold, followed by the average of all folds. The results here show both unique and non-unique performance through precision, recall and F-measure. We focus predominantly on increasing the F-measure for the unique mode of evaluation, and not for the non-unique or edge level performance.

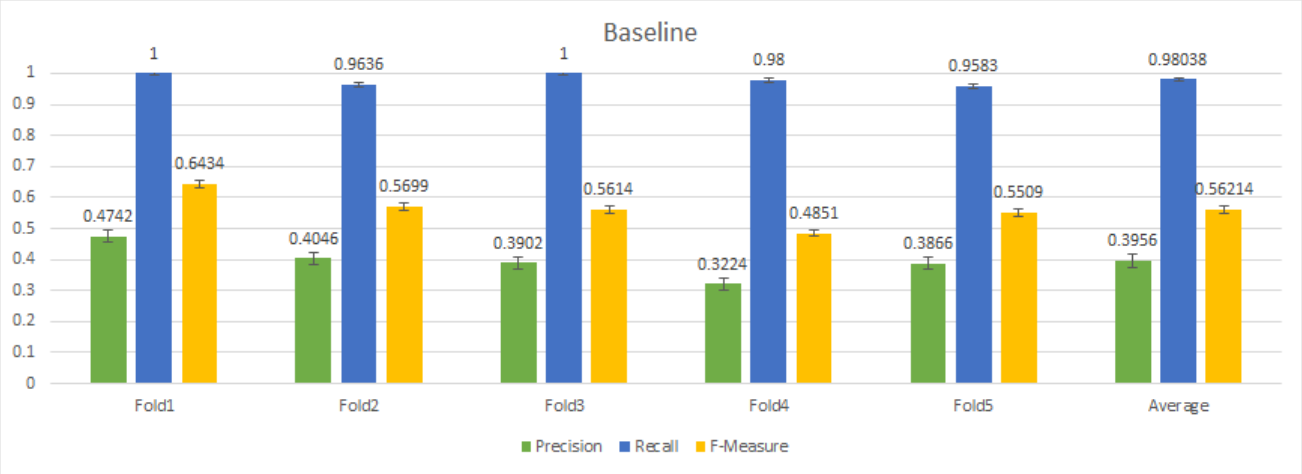

Figure 5.1: Document Level Statistics for Baseline Method.

5.1 Baseline Evaluation

To compare the performance of our methods, we choose an appropriate baseline method. This baseline is particularly important because there exists no other tool for text mining of transcription factor and protein relations. The baseline method is based on co-occurence in the same sentence. Any two entities that occur in the same sentence are reported as a relation. In the relna corpus, over 90% of the annotated relations are in the same sentence. Therefore, in the baseline method, the recall value is very close to 100%. However, as we can expect, less than 40% of the entity combinations in a

sentence actually are true relations. Therefore, the precision of the baseline method is only 39%. The average F-measure stands at 56.2%.

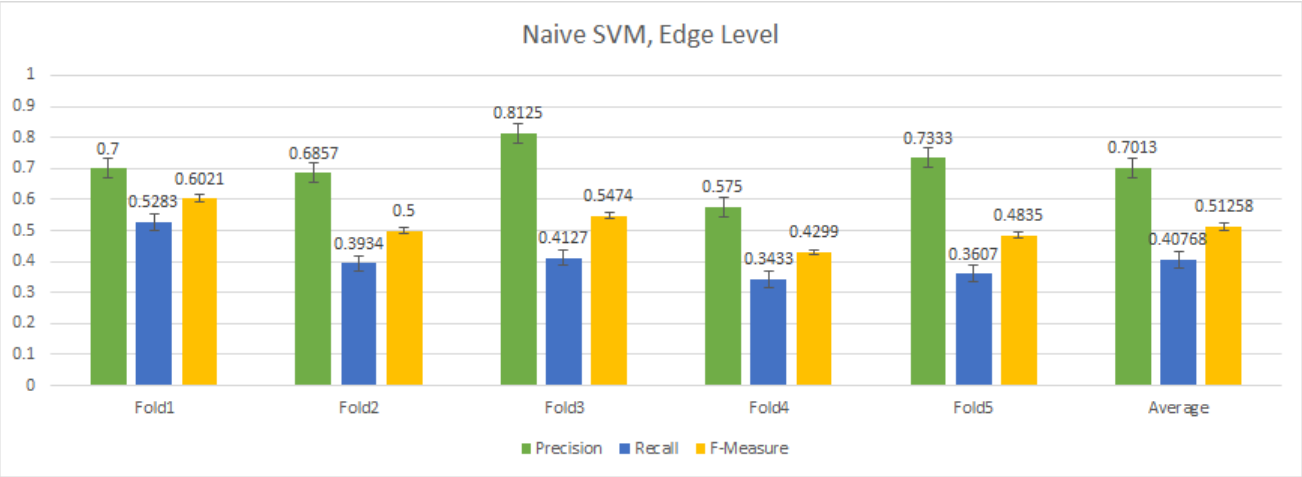

Figure 5.2: Edge Level Statistics for SVM with linear kernels.

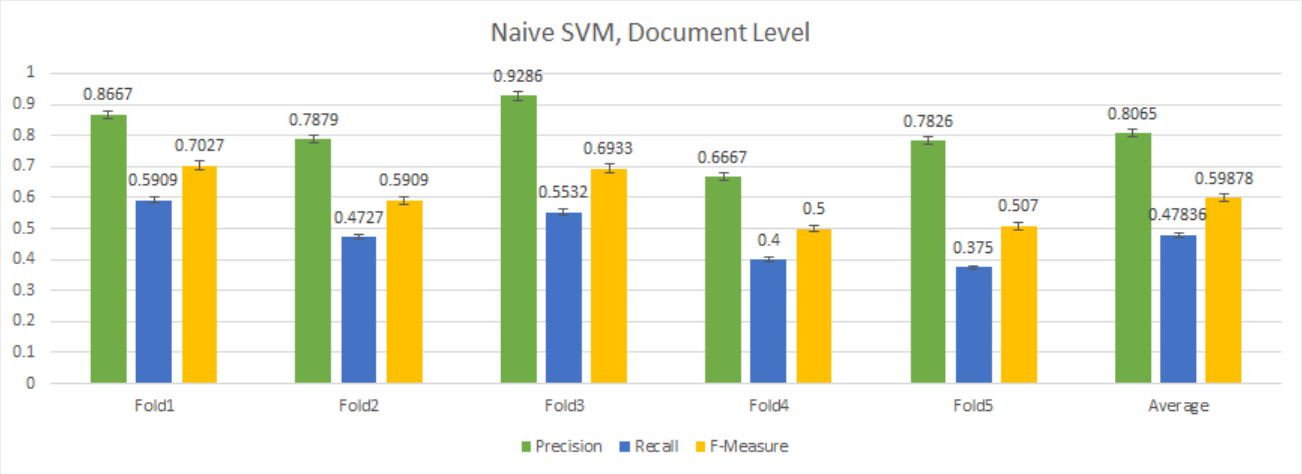

Figure 5.3: Document Level Statistics for SVM with linear kernels.

5.2 Naive SVM Evaluation

Naive SVMs form our initial results. We used SVMs with linear kernels. At the edge level, we report a precision of 70.1%, recall of 40.8% and F-measure of 51.3%. At the document level, we achieve a precision of 80.7%, recall of 47.8% and F-measure of 59.9%. Both of these results can be found in Fig 5.2 and Fig 5.3

5.3 SVM with Tree Kernels Evaluation

Subsequently, we tried SVMs with Tree Kernels. These compare the constituency parse trees. At the edge level, we report a precision of 86.6%, recall of 17.1% and F-measure of 28.0%. At the document level, we achieve a precision of 91.7%, recall of 21.4% and F-measure of 34.0%. Both of these results can be found in Fig 5.4 and Fig 5.5.

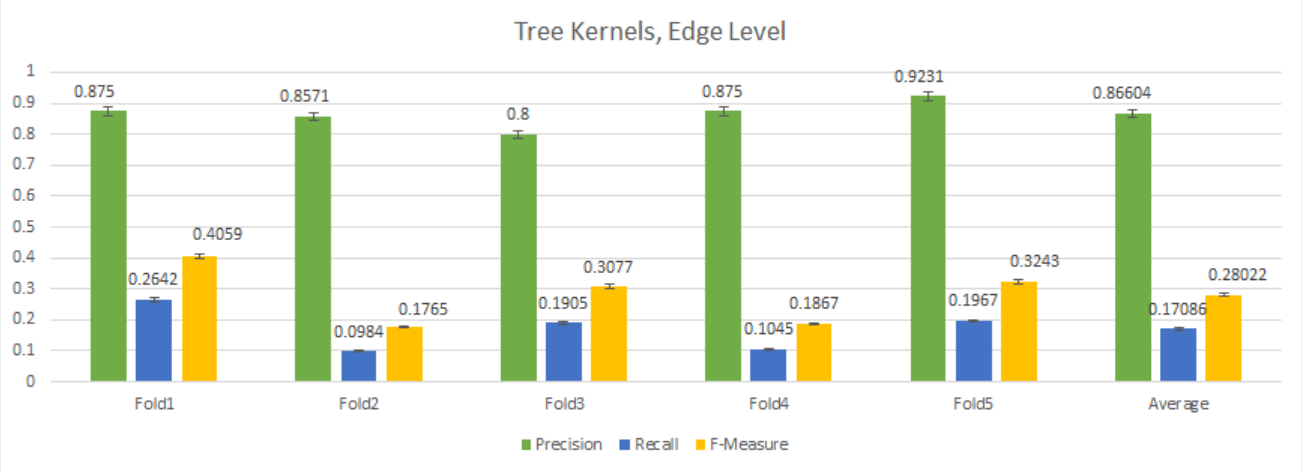

Figure 5.4: Edge Level Statistics for SVM with Tree Kernel

Even though the F-measure was low for this, Tree Kernels pose a promising

lead for extracting novel relations of transcription factors and their associated proteins because of the high precision.

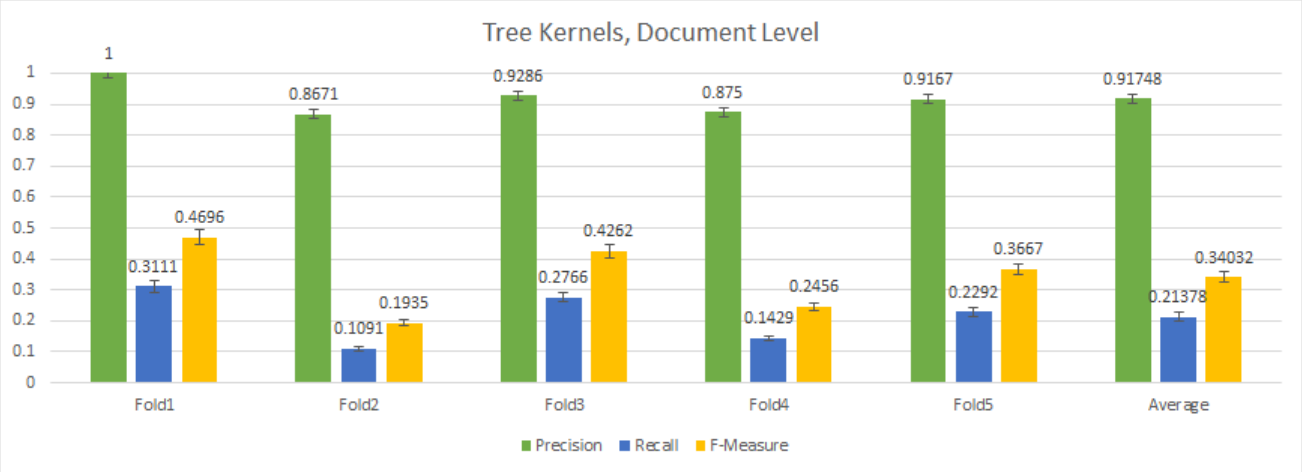

Figure 5.5: Document Level Statistics for SVM with Tree Kernel

5.4 SVM Classification after Correction of Class Imbalance

A major problem with the *relna* corpus was the imbalance of positive and negative classes. We observed that there were almost 4 times as many negative classes as positive classes, and therefore our model was being overfit to accommodate these negative classes.

To counter this, we employed a simple undersampling technique. Undersampling involves bringing the ratio of positive and negative classes to nearly equal values. We empirically reached an undersampling threshold of 0.4, i.e., only 40% of the negative classes were used to train the model. Using the default linear kernel of the SVM, we achieved a precision of 54.3%, recall of 58.0% and F-measure of 55.6% at the edge level. At the document level, we achieve a precision of 71.9%, recall of 65.3% and

F-measure of 68.3%. Both of these results can be found in Fig 5.6 and Fig 5.7.

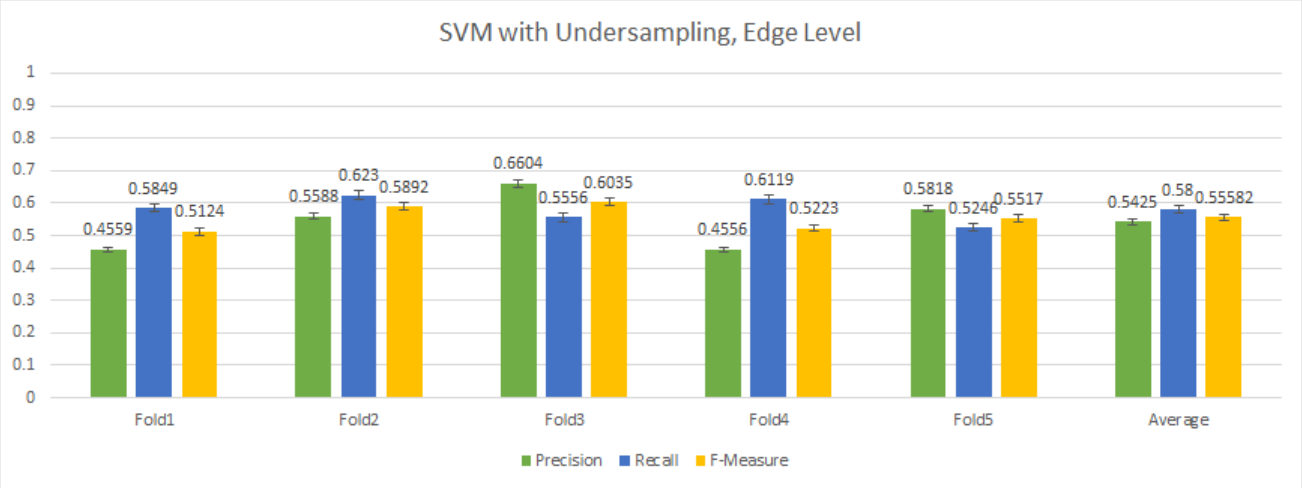

Figure 5.6: Edge Level Statistics for Undersampling of 0.4 with SVMs using Linear Kernel.

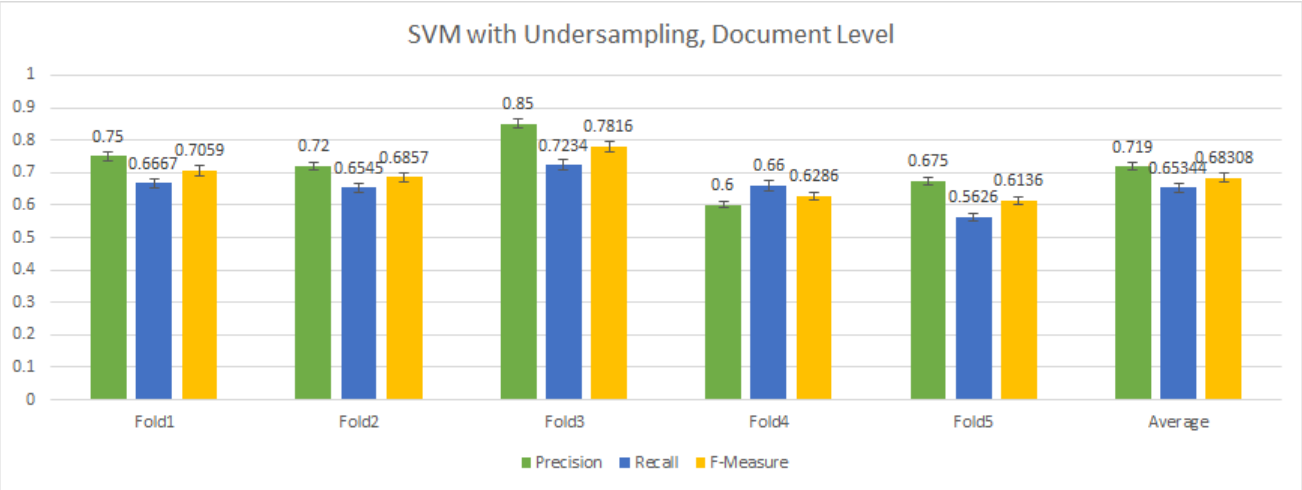

Figure 5.7: Document Level Statistics for Undersampling of 0.4 with SVMs using Linear Kernel.

This approach of undersampling provided the biggest gains in performance for the machine learning model. The F-measures increased by close to 10% from the naive SVM implementation.

5.5 SVM Classification after Hyperparameter Tuning

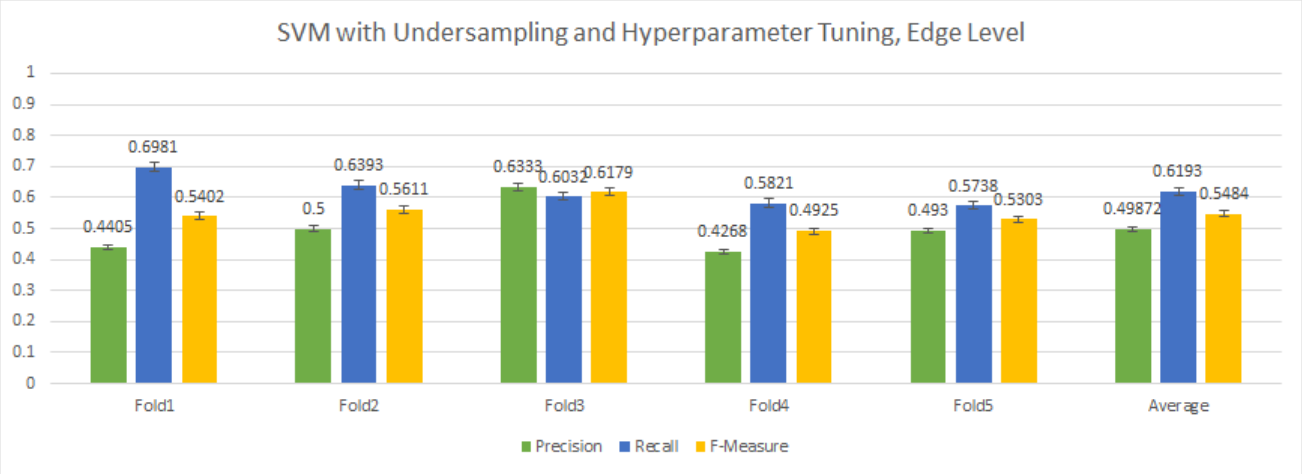

Figure 5.8: Edge Level Statistics for SVM with Linear Kernels with Undersampling and Hyperparameter Tuning.

In our final step, we optimized the C-parameter, the trade-off between the misclassification error and the margin of the hyperplane. At the edge level, we report a precision of 49.9%, recall of 61.9% and F-measure of 54.8%. At the document level, we achieve a precision of 69.1%, recall of 69.9% and F-measure of 69.3%. Both these results can be found in Fig 5.8 and Fig 5.9.

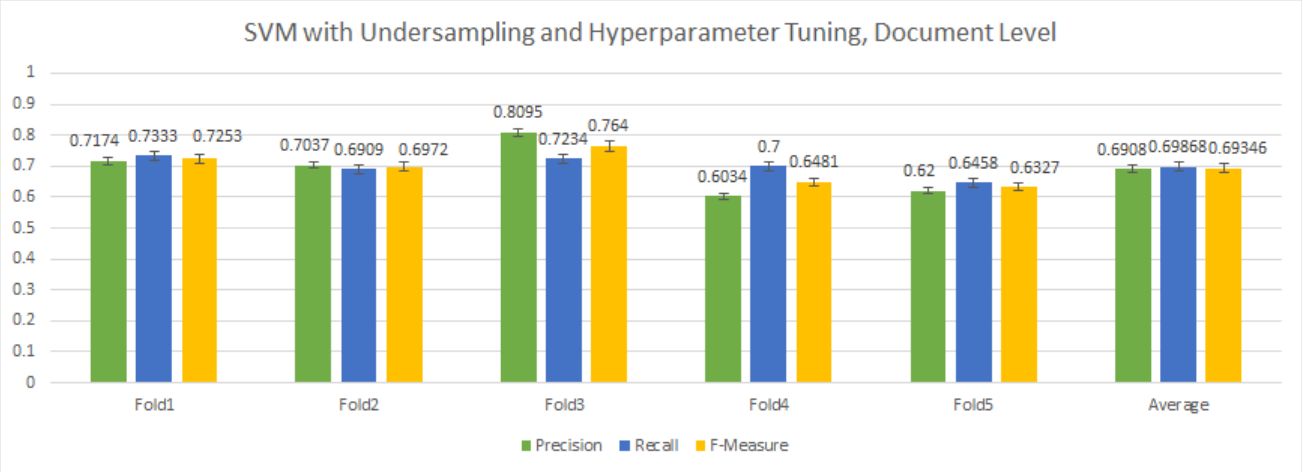

Figure 5.9: Document Level Statistics for Baseline Method.

5.6 Cost of Feature Selection

We also attempted to quantify the cost of feature selection in terms of performance (F-measure) and time-efficiency. In this experiment, we used the two kinds of splits (80:20 and 60:20:20) respectively. Feature selection was performed on the 60:20:20 split, where 20% of the dataset was used as a development set. Feature selection was performed using average information gain across groups of feature generators on this data to ensure that no bias carries forward to the training data.

We observed that the performance was, on an average, slightly higher without feature selection. However, this also meant that the time taken to train the model and classify new instances was slightly higher. This is because of the high dimensionality of the feature set. We did not find this to be a significant problem, and opted to use the model with higher performance.

Figure 5.10: The performance vs. time-efficiency graphs for models with and without feature selection.

6 Conclusion

This thesis focused on a novel method to extract relations of transcription factors and associated genes and gene products from biomedical literature. Since no corpora existed before for this purpose, I created my own corpus, *relna* and annotated manually relations between transcription factors and GGPs. The task of relation extraction is modeled as binary classification and draws heavily from concepts in NLP for creating a comprehensive feature set. This chapter details the most significant conclusions from this thesis.

6.1 Relna Corpus

Relna is the first corpus that annotates transcription factors and associated genes and gene products. We created this corpus semi-automatically. We used GNormPlus for entity recognition, and filtered the GGPs based on their GO Terms. This way, we were able to tag entities automatically. Moreover, all these entities are normalized to their Entrez Gene IDs and Uniprot IDs. *relna* consists of 150 abstracts.

The corpus covers around 300 relations, at an average of about 1.75 relations per abstract. Interestingly, most relations (over 90%) are contained in the same sentence. Most importantly however, we would like to report the success of a semi-automatic

approach to corpus construction that can be used to quickly develop new corpora faster. Since entity taggers are typically around 80% accurate, this gave us a strong initial base and reduced the effort in annotating and normalizing entities.

Link for corpus: *relna*: `http://pubannotation.org/projects/relna`

6.2 Relation Extraction Framework

Relna is a novel method of extracting relations of transcription factors and their associated genes or gene products, to the best of our knowledge. The task of relation extraction is modeled as binary classification using SVMs. We focused on extracting relations from the same sentence as they covered the bulk of relations from our corpus. Our features draw extensively from background in NLP, and specifically use dependency parse trees and constituency parse trees for classification. The dependency parse trees for each sentence is modeled as a bidirectional graph, and is used to build paths between entities. Properties of this path, such as the length, the direction of traversal, the dependencies along the path and the words in the path influence our classification task immensely.

We use 5-fold cross validation and comparison with a baseline for evaluations of our methods. The baseline method simply predicts any two entities in the same sentence as a relation. This has a precision of about 40%, recall of close to 100% and an F-measure of about 56%. Using linear kernels in SVM increased the performance to 59% of F-measure. After hyperparameter tuning and undersampling, we achieve an F-measure of 69.3%.

An interesting aspect of our experiments with SVMs used Tree Kernels. Using Tree Kernels increased the precision to over 91%. Even though the recall is low at

21%, this method could be used to identify potentially new relations in biomedical literature. Even if only a fraction of new relations were extracted, this would still be a large number.

6.3 Integration into nalaf

We built this relation extraction tool using *nalaf*, a framework for entity recognition and relation extraction. Our method uses several generic features that can indeed be used for any relation extraction task. Therefore, we integrated several of these features into the framework directly. Both *relna* and *nalaf* publicly available on the Rostlab GitHub page and can be run easily for any named entity or relation extraction task.

Links for the Project:

1. *relna*: `https://github.com/Rostlab/relna`
2. *nalaf*: `https://github.com/Rostlab/nalaf`

6.4 Future Scope

We believe neural networks hold promise for relation extraction from natural text. Nguyen et al [16] and Zeng et al [23] report the success of convolutional deep neural networks in relation extraction. We aim to integrate the open source libraries TensorFlow or Theano to our framework, and test our results again with neural nets.

We also look at a bootstrapping method for training and evaluating our method. Bootstrapping involves iterations of tagging and prediction, and allows our model to learn continuously. Currently, this approach is being used as part of *nala* for

extracting mutations. Another important aspect of generalizing relation extraction is to span multiple sentences, and look at relations at distances of one sentence and beyond.

List of Figures

List of Tables

Bibliography

[1] *AmiGO 2: Term Details for "transcription factor activity, sequence-specific DNA binding" (GO:0003700)*. `http://amigo.geneontology.org/amigo/term/GO:0003700`. (Visited on 11/29/2015).

[2] A. Bies, M. Ferguson, K. Katz, R. MacIntyre, V. Tredinnick, G. Kim, M. A. Marcinkiewicz, and B. Schasberger. "Bracketing guidelines for Treebank II style Penn Treebank project." In: *University of Pennsylvania* 97 (1995), p. 100.

[3] A. Borthwick, J. Sterling, E. Agichtein, and R. Grishman. "NYU: Description of the MENE Named Entity System as Used in MUC-7." In: *In Proceedings of the Seventh Message Understanding Conference (MUC-7*. 1998.

[4] J. M. Cejuela, P. McQuilton, L. Ponting, S. J. Marygold, R. Stefancsik, G. H. Millburn, B. Rost, F. Consortium, et al. "tagtog: interactive and text-mining-assisted annotation of gene mentions in PLOS full-text articles." In: *Database* 2014 (2014), bau033.

[5] E. Charniak and M. Johnson. "Coarse-to-fine n-best parsing and MaxEnt discriminative reranking." In: *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics. 2005, pp. 173–180.

[6] T. Joachims. *Learning to classify text using support vector machines: Methods, theory and algorithms*. Kluwer Academic Publishers, 2002.

[7] J.-D. Kim, T. Ohta, Y. Tateisi, and J. Tsujii. "GENIA corpus—a semantically annotated corpus for bio-textmining." In: *Bioinformatics* 19.suppl 1 (2003), pp. i180–i182.

[8] J.-D. Kim, T. Ohta, Y. Tsuruoka, Y. Tateisi, and N. Collier. "Introduction to the bio-entity recognition task at JNLPBA." In: *Proceedings of the international joint workshop on natural language processing in biomedicine and its applications*. Association for Computational Linguistics. 2004, pp. 70–75.

[9] T. A. Libermann and L. F. Zerbini. "Targeting transcription factors for cancer gene therapy." In: *Current gene therapy* 6.1 (2006), pp. 17–33.

[10] Y. Liu, Z. Shi, and A. Sarkar. "Exploiting rich syntactic information for relation extraction from biomedical articles." In: *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Companion Volume, Short Papers*. Association for Computational Linguistics. 2007, pp. 97–100.

[11] A. Moschitti. "Making Tree Kernels Practical for Natural Language Learning." In: *EACL*. Vol. 113. 120. 2006, p. 24.

[12] D. Nadeau and S. Sekine. "A survey of named entity recognition and classification." In: *Lingvisticae Investigationes* 30.1 (2007), pp. 3–26.

[13] *NCBI Gene Database*. `http://www.ncbi.nlm.nih.gov/gene/`. (Visited on 11/29/2015). 2015.

[14] *NCBI Text Mining Tools*. `http://www.ncbi.nlm.nih.gov/CBBresearch/Lu/Demo/tmTools/`. (Visited on 11/29/2015). 2015.

[15] *NCBI Text Mining Tools - Input and Output Formats*. `http://www.ncbi.nlm.nih.gov/CBBresearch/Lu/Demo/tmTools/Format.html`. (Visited on 11/29/2015). 2015.

[16] T. H. Nguyen and R. Grishman. "Relation Extraction: Perspective from Convolutional Neural Networks." In: ().

[17] N. Papanikolaou, G. A. Pavlopoulos, T. Theodosiou, and I. Iliopoulos. "Protein–protein interaction predictions using text mining methods." In: *Methods* 74 (2015), pp. 47–53.

[18] B. Percha, Y. Garten, R. B. Altman, et al. "Discovery and explanation of drug-drug interactions via text mining." In: *Pac Symp Biocomput*. Vol. 410. World Scientific. 2012, p. 421.

[19] L. Tanabe, N. Xie, L. H. Thom, W. Matten, and W. J. Wilbur. "GENETAG: a tagged corpus for gene/protein named entity recognition." In: *BMC bioinformatics* 6.Suppl 1 (2005), S3.

[20] P. Warrer, E. H. Hansen, L. Juhl-Jensen, and L. Aagaard. "Using text-mining techniques in electronic patient records to identify ADRs from medicine use." In: *British journal of clinical pharmacology* 73.5 (2012), pp. 674–684.

[21] C.-H. Wei, H.-Y. Kao, and Z. Lu. "GNormPlus: An Integrative Approach for Tagging Genes, Gene Families, and Protein Domains." In: *BioMed Research International* 2015 (2015).

[22] J. Wilbur, L. Smith, and L. Tanabe. "Biocreative 2. gene mention task." In: *Proceedings of Second BioCreative Challenge Evaluation Workshop*. 2007, pp. 7–16.

[23] D. Zeng, K. Liu, S. Lai, G. Zhou, and J. Zhao. "Relation classification via convolutional deep neural network." In: *Proceedings of COLING*. 2014, pp. 2335–2344.

[24] D. Zhou, D. Zhong, and Y. He. "Biomedical Relation Extraction: From Binary to Complex." In: *Computational and mathematical methods in medicine* 2014 (2014).