# A Multi-Level Boundary Classification Approach to Information Extraction

Aidan Finn

# Contents

# List of Figures

**Abstract**

Information Extraction (IE) is the process of identifying a set of pre-defined relevant items in text documents. We investigate the application of Machine Learning classification techniques to the problem of Information Extraction. In particular we use Support Vector Machines and several different feature-sets to build a set of classifiers for Information Extraction (IE). We show that this approach is competitive with current state-of-the-art Information Extraction algorithms based on specialized learning algorithms. We investigate the different components of our IE system, such as learning algorithm, feature-set and instance selection, and compare how much each component contributes to performance. We also introduce a new multi-level classification technique for improving the recall of IE systems. We show that this can give significant improvement in the performance of our IE system and gives a system with both high precision and high recall. Our system (ELIE) is an adaptive Information Extraction algorithm that uses a two-level boundary classification approach to learning. ELIE first classifies every document position as the start of a fragment to be extracted, the end of a fragment, or neither. This first level of extraction typically has high precision but mediocre recall. To increase recall, we employ a second level of classification. Positions near those positions extracted at the first level are classified by a second pair of classifiers that are biased for high recall. For example, the positions "downstream" from each extracted start position are classified in order to find the end of the given fragment. Our results on several benchmark corpora indicate that ELIE often outperforms state-of-the-art competitors.

# Chapter 1

# Introduction

## 1.1 Motivation

There are a huge number of electronic documents in the world today. Many are available on the world wide web while many others exist within organizations. These documents are often in unstructured or semi-structured format. Email and text documents often have little structure or some arbitrary structure that is defined by the document's author.

Information Extraction (IE) is the process of identifying a set of pre-defined relevant items in text documents.

IE has many applications. It can improve information retrieval and text mining by identifying entities in free text. It can extract structured data from unstructured text to create structured databases. It can be used to automatically add semantic annotation to web-pages.

It enables structured data to be built from unstructured or semi-structured text sources. It can be used to create a structured database from unstructured text. For example, a company may receive hundreds of resumes by email from potential employees. These will all be in some kind of structured textual format but this format will vary from resume to resume. It is difficult to compare and form complex queries over all the resumes in their textual form. IE techniques can be used to extract the relevant items from the text documents and construct a relational database from the data. This allows for complex queries to be constructed over

all the resumes e.g. “Find me all the applicants that have more than 3 years experience, are under the age of 25 and have listed java-programming as a skill”. It would be very difficult to perform such a search using keyword matching over the raw text of the resumes. IE can automatically identify and extract the various elements of the text that we are interested in.

The huge number of documents on the world-wide-web exhibit similar problems when it comes to formulating complex queries. Current search systems such as Google excel at retrieving documents according to what keywords they match but they do not allow for more complex queries. Current search methods see web documents as sequences of tokens. Searching consists of matching documents which contain the same tokens as the query and then ranking them according to some ranking metric such as the number of links that point to them. The semantic web initiative aims to address this problem by adding semantic mark-up to documents when they are created. This semantic information will facilitate more complex searching of documents on the web. However the vast majority of documents on the web contain no semantic mark-up. IE can be used to identify semantic entities in text and web documents. IE can facilitate the semantic web by automating the process of adding semantic mark-up to documents.

Information extraction can be done manually by having an expert user create rules that will extract the desired entities. This is a difficult, expensive and time-consuming process.

We are interested in Automated Information Extraction. Figure 1.1 shows what we mean by automated IE. A human annotator annotates examples of the entities that we want to extract. These annotated documents are used as examples for a Machine Learning algorithm. It uses these labelled examples to learn a set of rules that can be used to extract the entities. Thus the role of the human is reduced to labelling example entities in documents rather than having to construct complex sets of rules. The complexity of identifying the best rules is left to the learning algorithm.

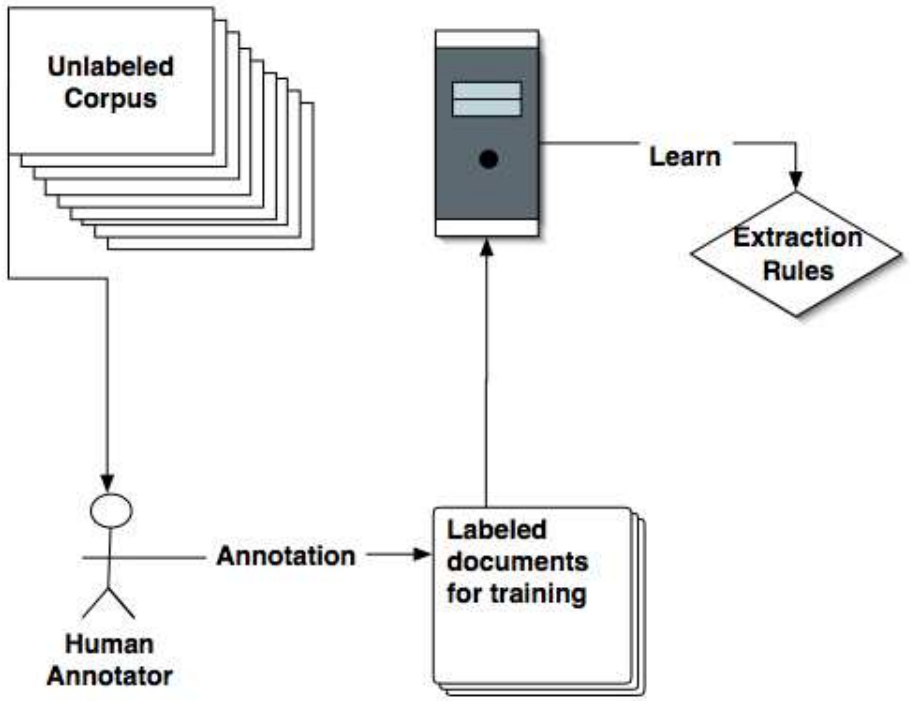


Figure 1.1: Automated Information Extraction

## 1.2 Contributions

Numerous IE systems based on Machine Learning techniques have been proposed recently. Many of these algorithms are "monolithic" in the sense that there is no clean separation between the learning algorithm and the features used for learning. Furthermore, many of the proposed algorithms effectively reinvent some aspects of Machine Learning rather than exploit existing Machine Learning algorithms. It is not obvious which aspects of each system contribute to its performance and how much each contributes.

We investigate how relatively "standard" Machine Learning techniques can be applied to Information Extraction. We adopt the standard "IE as classification" formalization [20, 12], in which IE becomes the task of classifying every document position as either the start of a field to extract, the end of a field, or neither. We then break down the different components of this system and investigate how each component contributes to and affects the performance. We investigate

attribute filtering, the effect of different feature-sets and the effect of choice of learning algorithm. We investigate the problems caused by data imbalance on the IE task and investigate different instance-undersampling strategies.

Based on this initial system, we then describe enhancements to this basic approach that give higher performance on a variety of benchmark IE tasks. Our enhancements consist of combining the predictions of two sets of classifiers, one set with high precision and one with high recall.

The intuition behind this approach is as follows: We assume the base classifiers have high precision and predict very few false positives. To extract a fragment we need to identify both its start and end tags. If the base classifier predicts one tag (start or end) but not the other we assume that it is correct. We use this prediction as a guide to a second classifier to try and identify the missing complementary tag.

We make three contributions. First, we show that the use of an off-the-shelf support vector machine implementation is competitive with current IE systems based on specialized learning algorithms. Second we investigate the effects of each component of the system (features, instances, learner) on the overall performance. Third, we introduce a novel multi-level boundary classification approach and demonstrate that this new approach outperforms current systems.

## 1.3 Organization

The rest of this thesis proceeds as follows. Chapter 2 describes some background in the area of Machine Learning and applying it to text. It describes some of the other state of the art IE systems.

Chapter 3 describes the standard benchmark datasets that are commonly used to evaluate IE tasks. We describe each of the datasets used. We also discuss the issue of how to evaluate an IE system. We describe some of the shortcomings of how previous systems have been evaluated and discuss how IE systems should correctly be evaluated. We evaluate our own system using a conservative methodology for comparison to other IE systems and we describe this methodology here.

Chapter 4 describes our basic approach that treats IE as a token classification task. This approach uses a generic Machine Learning implementation and a fea-

ture representation that represents the relational nature of the IE task. We show that this approach is competitive with other state of the art IE systems that use specialized learning algorithms.

In chapter 5 we delve into this system further and investigate which parts of it contribute to performance. We investigate the effect of various features-sets that our IE system uses. We investigate the effect of varying the learning algorithm. We also investigate the effect of filtering the attributes from our representation.

In chapter 6 we investigate the effect of instance filtering on our IE system. Because we are using SVMs for learning we only need the instances that make up the support vectors for learning. Our representation results in a large number of instances but in fact we can filter the majority of them without significantly affecting performance. We investigate two approaches to filtering negative instances. The first selects random negative instances for filtering. The second approach filters instances in a more focused manner. It filters negative instances that contain tokens that are unlikely to occur in the positive instances. These instances are also unlikely to be part of the support vectors.

Chapter 7 extends the IE system that we have developed in the preceding chapters. It adds a second level of classifiers. This second level of classifiers is designed to have high recall. It is combined with the first set of classifiers which have high precision and it uses their predictions to filter its own predictions. This produces a new set of predictions that have higher recall but maintain high precision. We show experimentally that this approach gives significant improvements in performance.

Chapter 8 analyzes and discusses our IE system's performance on the Pascal challenge, a recent challenge task for Information Extraction.

Chapter 9 further generalizes our two-level approach and addresses the problem of representing relational information between different fields. It adds a third level of classifiers. This third level aims to use relations between fields to improve performance. This level uses the predictions for all the different fields to add features to the representation that encode relationships between fields and then relearns the models using these new features.

Chapter 10 presents our conclusions and suggests directions for future work.

# Chapter 2

# Background and Related Work

## 2.1 Overview

In this chapter we describe the work related to our research and background from related fields.

We give an overview of Machine Learning and describe some Machine Learning algorithms. We describe previous work in Text Classification and how the text classification problem is represented for Machine Learning.

We describe the Support Vector Machines algorithm in detail and in particular we discuss Support Minimal Optimization (SMO). SMO is specialization of the SVM algorithm which we use for learning the classifiers that our IE system uses. We describe the operation of several other well-known IE algorithms.

## 2.2 Machine Learning

In this section we present an overview of the Machine Learning approach to automatically building classifiers. We describe the Machine Learning algorithms that we used for our experiments.

### 2.2.1 The Learning Process

Machine Learning involves learning from examples. It uses a set of training examples to extrapolate and learn patterns, and to learn correlations between the features that are used to represent each example and some specified concept. Machine Learning has been widely used for document classification and many other applications. Document classification is of particular interest to us as many of the techniques can be adapted to IE. The Machine Learning approach to document classification takes a set of pre-classified examples and uses these to induce a model which can be used to classify future instances. The classifier model is automatically induced by examination of the training examples. The human effort in this process is in assembling the labelled examples and choosing a representation for the training examples. A human must initially decide what features will be used to describe the training examples, and represent the training documents with respect to these features.

When using Machine Learning algorithms, we first identify the concept to be learned. This concept is what we want the classifier to be able to classify; in our case this is whether a token is a start or an end of a field.

The type of learning we are interested in is classification learning. In this learning scheme, the learner takes a set of labelled pre-classified examples. The learner is then expected to induce ways of classifying unseen examples based on the pre-classified examples given. This form of learning is supervised in that the training examples are provided and labelled by a human overseer.

The training data is a set of instances. Each instance is a single example of the concept to be learned. Instances are characterized by a set of attributes where each attribute measures a certain aspect of the concept being described. Attributes can be discrete or continuous. Continuous attributes represent some numerical value that can be measured. Discrete attributes assign the attribute to membership of a particular category.

Figure 2.1 shows an example Machine Learning dataset. The dataset contains 24 instances and 4 attributes. Each of the attributes is discrete i.e. they can take only certain pre-defined values. The attributes are:

1. Age of the patient.

| Age | Spectacle Pre | Ast | tear-pr-rt | lenses |
|---|---|---|---|---|
| young | myope | no | reduced | none |
| young | myope | no | normal | soft |
| young | myope | yes | reduced | none |
| young | myope | yes | normal | hard |
| young | hypermetrope | no | reduced | none |
| young | hypermetrope | no | normal | soft |
| young | hypermetrope | yes | reduced | none |
| young | hypermetrope | yes | normal | hard |
| pre-presbyopic | myope | no | reduced | none |
| pre-presbyopic | myope | no | normal | soft |
| pre-presbyopic | myope | yes | reduced | none |
| pre-presbyopic | myope | yes | normal | hard |
| pre-presbyopic | hypermetrope | no | reduced | none |
| pre-presbyopic | hypermetrope | no | normal | soft |
| pre-presbyopic | hypermetrope | yes | reduced | none |
| pre-presbyopic | hypermetrope | yes | normal | none |
| presbyopic | myope | no | reduced | none |
| presbyopic | myope | no | normal | none |
| presbyopic | myope | yes | reduced | none |
| presbyopic | myope | yes | normal | hard |
| presbyopic | hypermetrope | no | reduced | none |
| presbyopic | hypermetrope | no | normal | soft |
| presbyopic | hypermetrope | yes | reduced | none |
| presbyopic | hypermetrope | yes | normal | none |

Figure 2.1: Contact lens data: An example Machine Learning task

2. Spectacle prescription.

3. Whether the patient is Astigmatic.

4. The patient's tear production rate.

The class to be predicted in this case is whether the patient should have hard contact lenses, soft contact lenses or no contact lenses. Each instance is an example of a patient. Each patient is described in terms of the 4 attributes and whether or not

they received contact lenses. The task of a Machine Learning algorithm is to look at the example instances and try to learn patterns that will predict whether future patients will need contact lenses based on the attribute values for that patient.

This task is a simple illustrative task. The example instances contain all possible combinations of attributes and values, making it easy for a learning algorithm to learn rules that make predictions for future instances. This dataset can be covered with a rule-set consisting of only nine rules.

This dataset is very simple in comparison to most Machine Learning tasks. It has a small number of attributes and it has an example instance for every possible attribute-value combination. The examples can be covered with a small set of rules. Our IE task is much more complex than this example dataset. It generally consists of tens of thousands of instances and attributes. It generally takes a large set of rules to cover the entire dataset and the number of rules and instances that would be required to enumerate all possible attribute-value combinations is prohibitive.

There are many other well known datasets that are used to evaluate Machine Learning research. These are collected in the UCI repository [17] and cover many different tasks and are of varying complexity. Examples include predicting whether a patient has breast-cancer, diabetes, heart-disease, hepatitis, thyroid-disease, how a person votes and whether a person has a good credit rating.

### 2.2.2 Some Machine Learning Algorithms

In this section we describe some well known Machine Learning algorithms. The selection described are widely used and cover different broad approaches to Machine Learning. Some of them are similar to approaches used by the IE systems that we describe in section 2.5. OneR is a simple approach that is useful as a baseline and gives an indication of the complexity of a task. Naive Bayes is a probabilistic approach based on Bayes' rule that is widely used for Text Classification. ID3 is a decision tree induction approach. Winnow is designed for datasets with large numbers of irrelevant attributes and is used by the Snow-IE system. Ripper is a rule induction algorithm that is broadly similar to the rule-indication algorithm used by RAPIER and $(LP)^2$.

```
For each attribute
    For each value of that attribute:
        Make a rule that assigns the most frequent class of
        this value to this value
    calculate the error-rate of each rule.
Choose the rule with the smallest error rate.
```

Figure 2.2: OneR: An algorithm for induction of one-level decision trees

OneR is a simple learning algorithm for generating one level decision trees. It is one of the most basic of Machine Learning algorithms but has been shown to do remarkably well on many common Machine Learning datasets.

OneR generates a set of rules that classify an example instance based on a single attribute. Each attribute generates a set of rules, one for each value of the attribute. The error rate is evaluated for each attributes rule-set and the rule-set with the lowest error rate is chosen. Figure 2.2 shows the OneR algorithm.

Despite its simplicity, OneR performs well on many classification tasks. Holte [23] showed that OneR does surprisingly well in comparison to many state of the art algorithms on several commonly used datasets. It is on average just a few percent less accurate, but generates substantially simpler models and smaller trees. OneR can be used as an approximation of the complexity of a learning task. If OneR performs well on a learning task it indicates that it is not a particularly complex task as there is a lot of information in single attributes and complex combining of attributes is not necessary to learn the task.

ID3 [39] is an algorithm for top-down induction of decision trees. In the resulting decision tree, each node corresponds to an attribute. Each arc from the node corresponds to a possible value of that attribute. Each leaf describes the expected value of the discrete attribute for an instance described by the path from the root node to that leaf.

Each node should be associated with the attribute that is most informative among those not yet considered on the path from root to node. The notion of entropy is used to measure how informative an attribute is. In particular a measure of Information Gain is used to choose the most informative attribute.

Figure 2.3 shows a simple decision tree induction algorithm similar to that used by ID3. It uses a measure of Information Gain to choose which attribute to

```
DecisionTreeInduce(attributes,instances):
    Select an attribute to place at the root node
    Make a branch for each possible value
    if all instances have the same classification: break
    For each branch:
        DecisionTreeInduce(attributes not used reaching this branch,
                           instances that actually reach this branch)
```

Figure 2.3: ID3: An algorithm for decision tree induction

split on. The Information Gain of each attribute is calculated and used to choose the attribute with the highest Information Gain.

Another simple method is to use all attributes to calculate the probability of an instance belonging to a category based on the observed training data. This technique is called Naive Bayes [32] and is based on Bayes' rule. It naively assumes independence between attributes. Despite this assumption, Naive Bayes works well in practice, particularly when used on datasets that have fairly independent attributes. Naive Bayes can rival or outperform more sophisticated algorithms on many datasets and has been widely used for Text Classification.

Winnow [33] is an algorithm that is designed to deal with large numbers of irrelevant attributes. It consists of a threshold and a set of weights for all the attributes. For an instance, it predicts that it is a member of the positive class if the sum of all the weights for the instance's attributes is greater than the threshold. Learning consists of setting the attribute weights and adjusting them to minimize error on the training set.

Ripper [14, 13] is a fast rule-learning algorithm. It learns a set of if-then rules. The left hand side of each rule is a set of conditions and the right hand side is a classification. Ripper builds an initial set of rules and then optimizes them a pre-defined number of times. It is a covering algorithm. It builds rules greedily, one at a time. After a rule is constructed each example that is covered by that training algorithm is removed from the training set. This process continues until each example in the dataset is covered or until new rules have a high error rate. Each rule is just a conjunction of features.

### 2.2.3 Support Vector Machines

Support Vector Machine (SVM) [15, 7] is a Machine Learning algorithm for binary classification. Given a set of example instances, each instance labelled as being a member of one class or the other, the SVM algorithm identifies a hyperplane that separates the two classes.

If each attribute is represented by $n$ attributes then each instance can be plotted by a point in an $n$-dimensional space. The hyperplane separates the instances in this space. For a two-dimensional space the hyperplane is a line, for a three dimensional space the hyperplane is a plane and so on for higher dimensional spaces.

SVMs create a maximum-margin hyperplane between two classes that lies in a transformed input space. It tries to maximize the distance of closest examples to the hyperplane (margin). The feature space is a non-linear map from the original input space, usually of much higher dimensionality than the original input space. In this way, non-linear classifiers can be created. If there exists no hyperplane that can split the positive and negative examples, the soft margin method will choose a hyperplane that splits the examples as cleanly as possible, while still maximizing the distance to the nearest cleanly split examples.

SVMs select a small number of important boundary instances from each class and attempts to find a function that linearly separates them. An important aspect of the SVM algorithm is the ability to seperate instances that are not linearly seperable. The SVM algorithm transforms the instance space, which may not be linearly separable, to a new instance space where the classes are linearly separable. They find the maximum margin hyperplane between two classes. It does this using the kernel trick. The kernel trick uses a non-linear kernel function to transform the instances to another instance space where they are linearly seperable.

Figure 2.4 shows a simple two-dimensional hyperplane. In this example each instance has only two attributes (x and y), so we can plot the instances on a 2D graph as shown. The task of the learning algorithm is to find a hyperplane that can separate the instances into the positive and negative classes. In a two-dimensional space the hyperplane is a line. The example shows several possible hyperplanes that can split these examples (the dashed lines). This example is linearly sep-

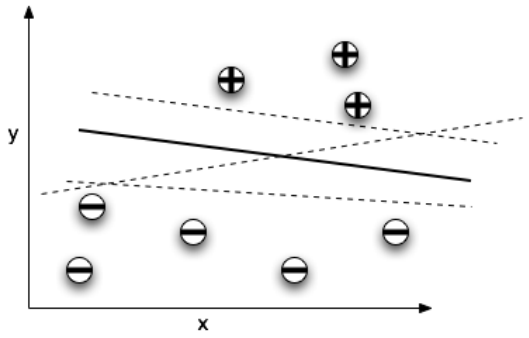


Figure 2.4: A hyperplane in two dimensions

arable in the original instance space, i.e. we can draw a line that separates the instances. When this is not the case, SVM maps the instances into another higher dimensional instance feature space where the instances are linearly separable. The SVM algorithm finds the maximal margin hyperplane (the solid line). The maximal margin hyperplane is the one that gives the greatest separation between classes - it is as far from each class as it can be.

The instances that are closest to the hyperplane are called the support vectors. There is at least one support vector for each class and usually more. The support vectors are sufficient to define the maximum margin hyperplane, i.e. we can construct the maximum margin hyperplane given only the support vectors. The instances that are not in the set of support vectors can be deleted or ignored without having any effect on the position or angle of the hyperplane. In the example of figure 2.4 the examples are linearly separable. If they are not the SVM algorithm can map them into another higher dimensional space where they are linearly separable (see figure 2.5).

SVM is currently one of the best learning algorithms for Text Classification [28] and has been successfully applied in many domains.

We use the Support Minimal Optimization (SMO) [38] algorithm. SMO is an SVM algorithm that is particularly suited for linear SVMs and sparse datasets. It exploits the sparseness of the data to improve performance. It replaces the quadratic programming inner loop of the SVM algorithm with a heuristic analytic quadratic programming step. It breaks down the quadratic programming problem

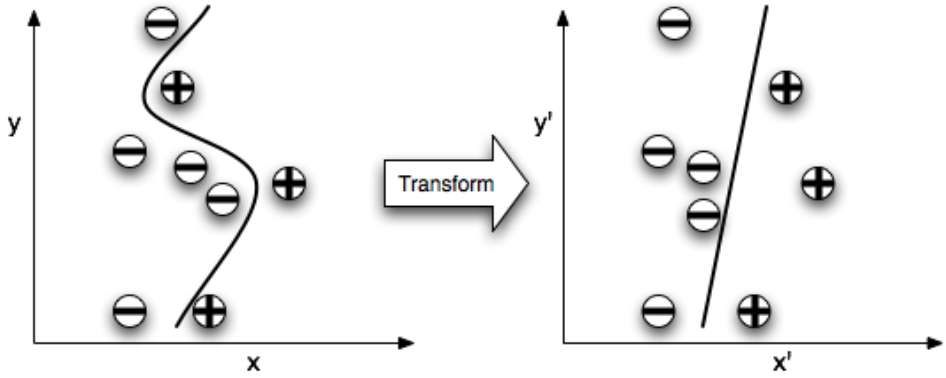


Figure 2.5: SVMs can map instances that are not linearly separable into another instance space where they are separable

into a series of smaller quadratic programming problems and at every step chooses to solve the smallest possible optimization problem.

## 2.3 Machine Learning and Text

Automatic Text Classification is the task of automatically assigning a document to one of a set of categories [44].

A typical application involves assigning documents to a pre-defined set of categories based on the topic of the document. An example of such an experiment is given in [27]. Data was collected from 20 newsgroups with 1000 documents collected from each. The task of the classifier was to automatically recognize which newsgroup a document came from. A Naive Bayes classifier achieved an average accuracy of 89% on this task.

There has been a large amount of work in the application of Machine Learning techniques to Text Classification (TC). The CORA system [34] is a hierarchical catalogue of computer science research papers. Documents are automatically spidered from the web and Machine Learning techniques are used to assign the document a place in the subject hierarchy. YahooPlanet [35] uses the Yahoo! hierarchy as a basis for automatic document categorization and applies Machine Learning techniques to the task of automatically assigning web documents to a category

within this hierarchy. Chen and Dumais [10] describe a system that automatically classifies search results into an existing category structure. Machine Learning approaches to text categorization have been successfully applied to the problem of spam filtering [43]. Spam filtering is the task of automatically recognizing and filtering spam email. Using Machine Learning for spam filtering is now a popular approach to the problem of automatic spam filtering. The Thunderbird email client uses a Naive Bayes classifier to filter junk mail. Apple's mail client uses Latent Semantic Analysis [16] for spam filtering.

When applying Machine Learning to text, the text is usually represented using the bag-of-words vector space model. Every token that occurs in all the training documents is a feature for the learner. Each document is represented as a binary vector of all these features.

TC and IE have some similarities but they differ in important ways. When using ML for TC, the task is to classify entire documents into categories. With IE, we are not classifying entire documents into classes. Instead we seek to identify fragments of documents that are of interest. With TC a single document is an example of a class but with IE a document may contain examples of several classes of interest. We are classifying the individual tokens of a document rather than the documents.

TC consists of a single classification task. Each model classifies a document as being a member of a class or not. IE consists of several classification tasks: for each token we must identify whether it is the start of a field and whether it is the end of a field. We must then combine the predictions of starts and ends to decide which combinations are fragments that should be extracted.

IE is a sequential task. With TC there it is assumed that there is no sequential relationship between the different documents being classified. With IE there is a strong relationship between each token being classified and the previous and next token being classified. We must extend the standard TC representations to take account of this sequential information. IE can be thought of as Text Classification at the token level with the additional need to represent sequence information between tokens. In traditional classification tasks such as TC, the classification of one object does not affect the classification of another, whereas in classification for IE, the class of one token depends on and affects the class of nearby tokens.

The large body of work in using ML for TC gives us a starting point in using ML for IE. We can reuse the learning algorithms and representations that have been widely used in the TC domain. We use similar representations for our instances but extend them to represent the sequential information between tokens. Existing IE systems are often monolithic systems that uses their own algorithms and representations for the IE task rather than taking advantage of the existing work in TC and ML. We can reuse much of the work that has been done in using ML for TC and apply it to IE with suitable modification.

## 2.4 The Semantic Web

The semantic web [4] is an extension to existing web standards that enables semantic information to be associated with web documents. The current www is designed for humans. It is not designed to be easily understandable by machines. The aim of the semantic web is to make web documents easier for machines to understand. It enables machine-readable information about the meaning of documents to be added to them.

For example, consider a page listing all the members of the Department of Computer Science at a University. Current search engines would treat it as a list of tokens with no information about what each of the tokens means. The semantic web enables us to encode extra meaning in the document about the meaning of the document and its various entities. For example, it would enable us to identify each entity listed as a person, and more specifically a computer scientist. Each person might have a name, web-site, email address and research interests associated with them. Using the semantic web enables machines to identify the concepts within a document. Suppose we wish to search for all computer scientists currently working in American universities in the area of artificial intelligence. Such precise queries are not possible using current search methods. If all web pages have semantic information associated with them this kind of query becomes easy.

The vast majority of current web pages have no semantic information associated with them. One of the barriers to the adoption of the semantic web is the difficulty in adding semantic annotations to large amounts of text. The ability to automatically add semantic annotations would be of huge benefit to adoption of

the semantic web [11]. IE is one process that can be used to bootstrap the semantic web by automatically identifying entities in existing web documents and using this information to add semantic annotations to the documents.

## 2.5 Information Extraction

In this section we discuss and compare some of the more prominent adaptive IE algorithms. These are the systems that we will compare our own against and are considered the state of the art in IE. They are the systems listed in the IE survey paper by Lavelli *et al*.[31].

### 2.5.1 Rapier

Rapier [8] uses inductive logic programming techniques to discover rules for extracting fields from documents. It does not try to identify start and end tags separately, but learns to identify relevant strings in their entirety. Rapier performs specific-to-general bottom-up search by starting with the most specific rule for each positive training example and repeatedly trying to generalize these rules to cover more positive examples. Rapier uses as its features the tokens, part-of-speech information and some semantic class information.

Rapier uses a different representation to the other IE systems. Rather than marking occurrences of fields in the text with an XML-like syntax, it takes a template filling approach. Associated with each text is a template containing the fields for that text. This approach does not distinguish between different occurrences of a field and doesn't allow for ambiguous text. For example, a job advertisement might have a template that contains 'platform:windows'. This approach doesn't allow us to represent the occurrence of the word 'windows' in the text in contexts other than platform. The task of the IE algorithm is to fill the template rather than to identify all occurrences of a field in the text.

Rapier's learning algorithm consists of specific to general search. It starts with the most specific rule-set for all the training examples and proceeds by replacing sets of rules with more general ones. The rules that Rapier learns consist of three parts: a pre-filler, a post-filler and a filler. The pre-filler matches the text before

the field, the post-filler matches the text after the field and the filler matches the actual field. Each pattern is a sequence of elements that that can be matched. As features, it uses only the actual tokens and their POS tags. For the initial rule-set the most specific rule for each example is created using the word and POS tags for the filler and its complete context. The initial rule-set is as specific as possible, consisting of a filler pattern of the exact tokens for the filler, a pre-filler pattern of all tokens that precede the filler in the document and a post-filler pattern of all the words that follow the filler in the document. Rapier then proceeds to generalize these rules by selecting pairs of rules and generalizing them by getting the least general generalization of each pair of rules. To consider all possible pre- and post-filler patterns would be prohibitive so Rapier starts generating pre- and post-fillers from the filler outwards. It maintains a list of the *k* best rules and repeatedly adds generalizations of the pre- and post-filler seed rules, working outward from the filler. The rules are ordered by Information Gain and weighted by the size of the rule, with small rules being preferred. When a rule gives no bad predictions on the training examples it is added to the final rule-base replacing any less general rules that it renders superfluous.

### 2.5.2 Boosted Wrapper Induction

Boosted Wrapper Induction (BWI) [20] learns a large number of simple wrapper patterns, and combines them using boosting. BWI learns separate models for identifying start and end tags and then uses a histogram of training fragment lengths to estimate the accuracy of pairing a given start and end tag. BWI learns to identify start and end tags using a form of specific-to-general search. BWI's features consist of the actual tokens, supplemented by a number of orthographic generalizations (alphabetic, capitalized, alphanumeric, lower-case, numeric, punctuation), as well as a modest amount of lexical knowledge (a list of first and last names).

Boosting is a technique for improving the performance of learning algorithms by repeatedly learning from the training examples, each time changing the weights associated with individual examples. Each round of boosting pays more attention to examples that were difficult for the previous round.

Boosting works by combining multiple models. Each of these models can make a prediction for a particular instance and the predictions of all the models are combined according to how accurate each model was on the training data. Each model is built based on the previous model and focuses on instances that were difficult for the previous model.

Other systems attempt to learn rules that cover as many examples as possible. BWI learns rules that are specific and have high precision and limited recall but learns lots of them. This "weak" learning algorithm is then improved using boosting. It is repeatedly applied to the training set and each time the weights associated with each example are changed to emphasize examples on which the learner has done poorly on previous steps.

BWI treats IE as a token classification task, where the task is to classify each token as being a boundary that marks the beginning or end of a field. It builds up a set of patterns from the training set. It builds two sets of patterns - one for detecting starts of boundaries and one for detecting ends of boundaries. When starts and ends are identified, a fragment is extracted based on the probability of a fragment of that length occurring.

### 2.5.3 $(LP)^2$

$(LP)^2$ [12] learns symbolic rules for identifying start and end tags. Like BWI, it identifies the starts and ends of fields separately. In addition to token and orthographic features, $(LP)^2$ uses some shallow linguistic information such as morphological and part-of-speech information. It also uses a user-defined dictionary or gazetteer. Its learning algorithm is a covering algorithm which starts with specific rules and tries to generalize them to cover as many positive examples as possible. This process is supplemented by learning correction rules that shift predicted tags in order to correct some errors that the learner makes.

It operates in two steps. The first step uses a simple bottom-up generalization process to learn a set of tagging rules. The second step learns a set of correction rules that correct errors made by the tagging rules.

The first step involves learning a set of tagging rules. Each rule attempts to identify either the start or the end of a fragment, rather than trying to recognize

whole fragments at once. $(LP)^2$ takes a classification approach to IE with each start or end of a fragment being a positive example and all other instances being negative examples.

It proceeds as follows: for each positive example: 1) build an initial rule, 2) generalize the rule, 3) keep the *k* best generalizations of the rule and discard the rest.

The generalization process consists of taking the specific initial rule and trying to generalize it with some information gained from shallow NLP analysis. For example 'at 3 pm' might be generalized to 'at [digit] pm'. This could be further generalized to 'at [digit] [timeid]'. Once $(LP)^2$ has generated all the generalizations, the next step is to select the best generalizations. Each generalization is tested on the training corpus. The *k* best generalizations are kept that 1) have better accuracy, 2) cover more positive examples, 3) cover different parts of the input and 4) have an error rate that is less than some specified threshold. Rules that are not discarded at this step are added to a best rules pool. Instances that are covered by a rule in the best rules pool are removed from the positive examples. Thus once an instance has been covered by a rule, it is no longer used for rule induction.

Once the initial set of tagging rules has been generated, $(LP)^2$ tries to learn contextual rules. The initial rule-set tends to have high precision but low recall. This phase attempts to increase recall by learning what are termed contextual rules. Some of the discarded rules are retrieved and $(LP)^2$ attempts to constrain their application to make them reliable. This constraint is derived by exploiting interdependencies among tags. The first rule pool was derived by assuming that all tags were independent. This is not always the case. For example, if we identify an *stime*, then an *etime* may follow. Some rules that were previously discarded are re-introduced with constraints if adding the constraint improves the rules performance. For example, A rule that identifies a start of *etime* might only apply if an end of *stime* has already been identified.

$(LP)^2$ also attempts to induce correction rules. These rules attempt to learn to correct mistakes. For example, If the system extracted 3 from the fragment 'at 3 pm', the system tries to learn a correction rule that would shift the end boundary to the correct token. These correction rules learn to shift the position of tags that have already been identified and they are learned using the same process as the

tagging rules.

The use of contextual rules means that $(LP)^2$ is strong for related fields as it can exploit the dependence between them. $(LP)^2$'s learning algorithm is fairly simple. It seems likely that it performance comes from the ability to generalize using NLP features and the ability to use information about other tags to exploit relationships between tags.

### 2.5.4 SNoW-IE

SNoW [41] is a relational learning algorithm that is specifically tailored towards large-scale learning tasks such as IE. SNoW-IE [42] is an IE system based on SNoW. It identifies fragments in their entirety rather than separately identifying start and end tags. It uses token, orthographic, POS and semantic features.

SNoW-IE learns in two stages. The first stage is a filtering stage. The set of all possible fragments is filtered to a small number. The aim is to filter out irrelevant negative instances without filtering positive instances. A fragment is filtered if it matches one of 2 criteria: 1) it doesn't contain a feature that is common in the positive examples or 2) the fragment's confidence value is below a certain threshold. The first stage has high recall, while the second stage has high precision.

SNoW-IE uses relational learning for IE. Specifically, it uses a restricted form of Inductive Logic Programming (ILP). This system does not treat identifying starts and ends as separate token classification tasks. It extracts fragments in their entirety. It proceeds by identifying all candidate fragments in a document. Each of these fragments is represented using a set of defined features. Features are extracted from three regions: the fragment itself, before the fragment and after the fragment.

The second stage involves picking the correct fragments from the fragments that remain. The second stage uses an enhanced representation to improve performance. The remaining fragments are used to train a classifier for each field.

### 2.5.5 Hidden Markov Models and Conditional Random Fields

A Hidden Markov Model (HMM) is a probabilistic finite state automaton for modelling sequential data. Each state omits tokens according to some fixed distribu-

tion. The transition between states also conforms to some fixed distribution. When using HMMs for IE, the states represent the fields to be extracted. For example, there would be states for the start and end of each field and a state for none of the fields (background state). There are efficient algorithms for generating the most likely state sequence for a given document. HMMs can be used for IE by determining the sequence of states that were most likely to have generated the entire document, and then extracting the symbols associated with the fields we wish to extract.

Freitag and McCallum [21] describe an approach to using HMMs for IE. The HMM models a generative process where a sequence of symbols is generated by starting at some start state, generating the symbol designated by that state and transitioning to the next state. The process of generating a symbol and transitioning to a new state is repeated until a final state is reached. Associated with each possible set of states is a probability distribution over all symbols in the vocabulary and a probability distribution over a set of transitions to the next state. These probabilities are learned from the training date. A dynamic programming algorithm called the Viterbi algorithm is used to find the most likely state sequence given a HMM and a sequence of symbols. They generate independent HMMs for each field to be extracted. Each model has two states - background state and target state. The target state produces the field that we wish to extract.

One of the weaknesses of using HMMs for IE is that it is difficult to model several different features for each token. As well as observing that a particular word occurred in the sequence, we may wish to observe that it was also capitalized and that it was the start of a noun phrase. Representing these extra features for the sequence is prohibitively difficult for HMMs.

Conditional Random Fields (CRFs) [29] are an approach that allow the use of arbitrary features in modelling the observed sequence. CRFs define a conditional probability distribution over labelled sequences rather than a joint distribution over pairs of labels and observation tokens. This allows the model to include arbitrary non-independent features as input.

To date CRFs haven't been extensively applied to the standard IE benchmark datasets but they have been shown to perform well on other IE tasks [37]. The ability to combine several different features for each token and probabilistically

model relationships between fields means that they are a very promising algorithm for IE.

## 2.6 Summary

This chapter covered background material and related research. We gave an overview of the Machine Learning process and described several Machine Learning algorithms. We described how Machine Learning is used for Text Classification. We also described several state-of-the-art IE algorithms: Boosted Wrapper Induction, Rapier, $(LP)^2$, SNoW-IE and Hidden Markov Models.

# Chapter 3

# Datasets and Evaluation for Information Extraction

There are several standard datasets that are used to evaluate the performance of Automated Information Extraction systems. These datasets are annotated by humans and the annotations are used as examples from which an IE system can learn generalization rules. In this chapter we will describe the datasets used for evaluation. We will also describe how we evaluate our system and discuss some of the issues surrounding the evaluation of IE systems. It is difficult to compare each of these systems directly for several reasons. First, there are a variety of "simple" methodological differences (e.g., using a slightly different scoring mechanism) [31]. More significantly, most of the literature reports only results for the system in its entirety. It is difficult to know how much of the performance differences arise from different feature sets, how much from different learning algorithms, and how much from differences in experimental procedure.

## 3.1 The Annotation Process

Before a dataset can be used for Automated Information Extraction it must be annotated. This involves identifying and marking all occurrences of the pre-defined fields that we wish to extract. All the datasets that we use for evaluation are already annotated and have been used widely by the IE community for evaluation.

However the annotation process itself is complex and deserves attention. All of the standard datasets contain annotation errors and inconsistencies. Annotation is the most time-consuming part of any new IE task.

The process of annotating documents for Automated Information Extraction involves identifying and marking the entities of interest in the text. This can be broken into several tasks. The first task is to identify what entities we are interested in extracting. The second task is to decide on a representation for the annotations. The third is to manually identify and mark the occurrence of the entities in the document corpus.

### 3.1.1 Defining what is to be Extracted

The first task is to identify and define the entities that we are interested in extracting from the document collection. Many of the entities that are of interest may be obvious from the dataset or from the particular extraction task. However it is important to define precisely what the entities are and what constitutes membership of a particular entity class. There may be some ambiguity as to how the entity is defined and how exactly to identify membership of the entity class. For example, in the Seminar Announcements dataset one of the fields of interest is *location*. This fields denotes the *location* of a particular seminar. There is some ambiguity in how *location*s are annotated however. If a document contains a string such as 'room B2.18, Computer Science Building, University College Dublin', should we annotate as a *location* 'room B2.18', 'Computer Science Building', 'room B2.18, Computer Science Building' or 'room B2.18, Computer Science Building, University College Dublin'?

Similarly, if we are to extract a time field, we must define what constitutes a time. Precise examples of times include '3:00', '3pm' and '3.p.m.'. We should define whether we allow more general time descriptions such as 'early afternoon', 'later today' and 'at the weekend'. Specifying in detail the fields that are to be extracted will result in higher annotation accuracy and less ambiguity in the annotations.

### 3.1.2 Annotation Syntax and Representation

The second task is to decide on a representation for the annotation. We are concerned with implicit relation extraction. The relations between different fields are implicit rather than explicit.

The simplest representation is to mark the start and end of each occurrence of a field in the text. Datasets are typically annotated for Information Extraction using a simple start and end syntax: *<field-name>This is a field</field-name>*.

This is the approach to annotations that our system used. It is also the approach that is used in most of the IE systems that we compare to and in the datasets that we use for evaluation.

This approach marks every occurrence of a field in a document. Another approach is the template mark-up approach. This approach was used by Rapier and was used to annotate the original Job Postings corpus. With this approach every document has an template associated with it which contains the values for each field. The template representation doesn't allow for occurrence of strings that match a target field in contexts outside of the target fields. As an example consider a job posting that has 'windows' as a required *platform*. The template associated with this job posting will specify 'windows' as the *platform*. However if the token 'windows' occurs in the document in contexts other than *platform*, this approach cannot represent that extra information. It must assume that all occurrences of the token 'windows' in the document are examples of the *platform* field.

These methods of annotation are limited in that they don't allow us to represent different forms of the same entity or relationships between entities. For example, if a document has *speaker*s 'Professor Sara Kiesler', 'Prof. Kiesler', 'Woody Vaskula', 'Vaskula' we cannot represent the fact that the first two strings refer to the same entity and the third and fourth refer to the same entity which is different to that referred to by the first two.

Figure 3.1 shows an example of the standard approach to annotation that is used for the benchmark IE datasets. It also gives an example of a slightly richer annotation scheme that allows us to associate annotations with entities.

The lack of richness in the standard annotation schema gives rise to problems

1. Standard approach to representation

Seminar: On evaluation of Information Extraction Systems
Speakers: <speaker>J. Doe<speaker> and <speaker>J. Bloggs</speaker>
Time: </stime>3:00 p.m</stime>

There will be a seminar by <speaker>John Doe</speaker> and <speaker>Jane Bloggs<speaker> at <stime>3pm</stime> in <location>room 101</location>. At <stime>3</stime>, <speaker>Jane<speaker> will speak followed by <speaker>John<speaker> at 3:30.

2. Representing entities

Seminar: On evaluation of Information Extraction Systems
Speakers: <speaker id=1>J. Doe</speaker> and <speaker id=2>J. Bloggs</speaker>
Time: <stime id=1>3:00 p.m</stime>

There will be a seminar by <speaker id=1>John Doe</speaker> and <speaker id=2>Jane Bloggs<speaker> at <stime id=1>3pm</stime> in <location id=1>room 101</location>. At <stime id=1>3</stime>, <speaker id=2>Jane</speaker> will speak followed by <speaker id=1>John</speaker> at 3:30.

Figure 3.1: Approaches to annotation for Automated Information Extraction

when it come to evaluating IE systems. Using the second method of annotation would allow us to perform a more correct form of evaluation. We will discuss these problems in more detail in section 3.3.

### 3.1.3 Annotating the Documents

The outcome of the processes described in the previous two sections is a set of annotation guidelines that are used to annotate the documents in the dataset. Even with concise and well defined guidelines annotation is a time consuming and error prone process. Human annotators can become tired and lose concentration when faced with a large annotation task. Document annotation is a tedious process and it is difficult for an annotator to completely avoid making errors.

There are some annotations that will not fall within the annotation guidelines and on which humans may disagree. One approach to this problem is to mark these annotations as optional or unsure. This is the approach taken in the MUC named entity recognition annotation scheme. This results in many spurious annotations as it allows the annotator to opt out of making a decision and annotate fragments that have only a tenuous connection to a field as described in the annotation guidelines.

Another approach is to have several annotators with overlapping sets of docu-

ments to annotate. If each document is annotated independently by two people we can assign extra annotators to documents where there was disagreement between the original two annotators. The level of agreement between annotators can serve as a useful measure of how well defined the annotation task is. If inter-annotated agreement is below a certain threshold, then the task may need to be reviewed and better defined.

## 3.2 Information Extraction Datasets

There are several benchmark datasets that are commonly used for Information Extraction. Each of these corpora were annotated by different independent researchers for evaluating their own particular system. In addition to these datasets, the Pascal Challenge dataset is a new dataset for IE evaluation. We describe the Pascal Challenge in a separate chapter (chapter 8) as it used different methodology and compared different systems.

We evaluate our system using these standard benchmark datasets: the Seminar Announcements (SA) dataset [19], the Job Postings (Jobs) dataset [8] and the Reuters Corporate Acquisitions (Reuters) dataset [18].

### 3.2.1 Seminar Announcements

The Seminar Announcements dataset consists of a set of 486 emails announcing seminars collected at Carnegie Mellon University. It is annotated for 4 fields: *speaker*, start-time (*stime*), end-time (*etime*) and *location*. The *stime*, *etime* and *location* fields may each occur several times in the Seminar Announcement and in different forms but they all refer to a single entity for the particular Seminar Announcement. i.e. a seminar can only have one *stime*, but this may occur several times in the document in different forms. e.g. '2:30', '230pm'. The *speaker* field can have multiple values as there can be more than one *speaker* at a seminar. However this information is external and is not encoded in the annotation. In fact it cannot be encoded in the current annotation scheme (see Figure 3.1).

Figure 3.2 shows an example Seminar Announcement. In this example all four fields are annotated. There are multiple *speaker*s for the seminar. This particu-

<0.16.1.95.11.46.22.copetas+@GANDALF.CS.CMU.EDU (Catherine Copetas).0>

Type: cmu.cs.scs

Topic: APPLE COMPUTER TALK

Dates: 26-Jan-95

Time: <stime>12:00</stime> - <etime>1:30 PM</etime>

PostedBy: copetas+ on 16-Jan-95 at 11:46 from GANDALF.CS.CMU.EDU (Catherine Copetas)

Abstract:

<speaker>Kai-Fu Lee</speaker> and <speaker>Rick Shriner</speaker> from Apple Computer will give a presentation on the Apple Core Technology group and the AdvancedTechnology Group on Thursday, January 26 from <stime>12:00</stime>-<etime>1:30 pm</etime> in<location>Wean 4625</location>. They will bring some of the latest Apple technologiesin communications, telephony, multimedia, and speech. Rick and Kai-Fu invite interested grads and undergrads to attend this informaldemonstration and discussion. Graduate students interested in potential job opportunities are particularly encouraged to attend. Pizza will be served.

Figure 3.2: An example Seminar Announcement

| Field | occurrences | Examples |
|---|---|---|
| *speaker* | 759 | Professor Sara Kiesler, Woody Vaskula, Vaskula |
| *stime* | 984 | 12:00 PM, noon, 5pm |
| *etime* | 435 | 1:30 PM, 1:30 p.m, 5pm |
| *location* | 645 | room 207, Student Activities Center, Baker Hall 355 |

Figure 3.3: Details of the Seminar Announcement dataset

lar example has very little structure but some of the other documents are more structured. The Seminar Announcements are all free text written as email and are thus generally unstructured but in some cases the author has composed the email in some structured form. This structure is dependent on the author and is not consistent across documents.

Figure 3.3 show details of field occurrences and examples from the Seminar Announcement dataset. The *speaker* is the name of the person giving the seminar. The *location* is where the seminar will take place. The *stime* field and *etime* are the time the seminar will start and end respectively. The *speaker* field is a multi-valued field while the others are single-valued. The most common field is *stime* while etime is the least common. Even so, it still occurs 435 across the dataset.

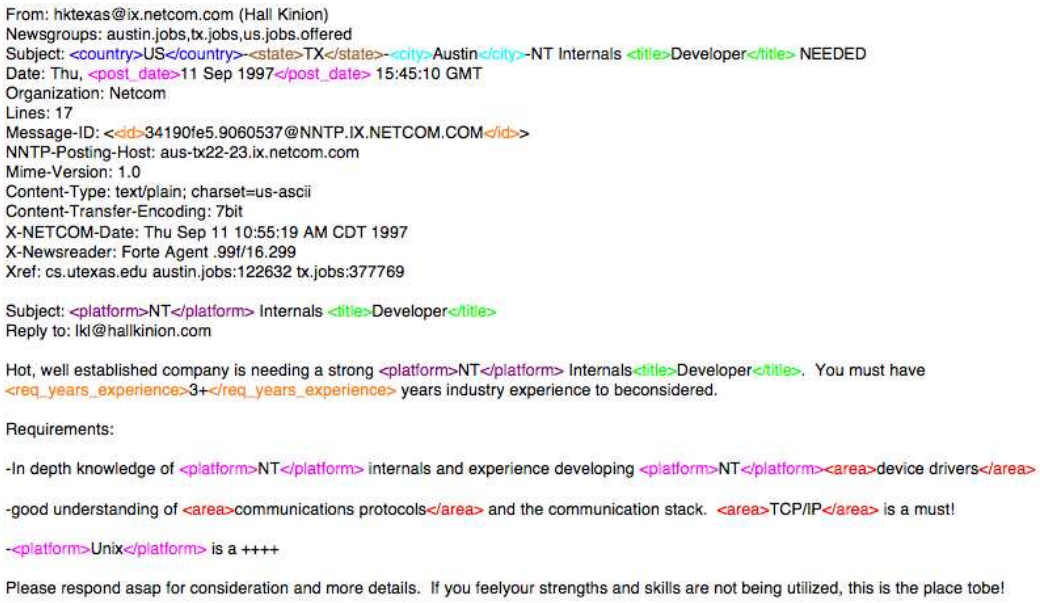


Figure 3.4: An example Job Posting

Most documents in the dataset have multiple occurrences of *speaker*, *stime* and *location*. Many documents contain all four fields.

We used the original version of the Seminar Announcement corpus. This corpus contains a large number of errors and inconsistencies. There is now a cleaned-up version of the corpus with many of the errors corrected. We use the original version for comparison with other systems that used that corpus. It is also debatable whether it is desirable to clean up the corpus. Real annotators make mistakes and by removing this noise from the annotation data we may be encouraging the learner to over-fit the corpus. If we remove all the errors from the data it could become artificial and may be less use for evaluating systems for real-world use.

### 3.2.2 Job Postings

The Job Postings dataset consists of 300 newsgroup messages detailing jobs available in the Austin area. Figure 3.4 shows an example Job Posting. This data is semi-structured. The first part of the message was created by the mailing program so it is strictly formatted. However the rest of the message was created by a hu-

| Field | Occurrences | Examples |
|---|---|---|
| *id* | 299 | NEWTNews.872347949.11738.consults@ws-n |
| *title* | 466 | ALC Application Programmer, Visual Basic Developers |
| *company* | 291 | Alliance, CPS, Charter Professional Services Inc |
| *salary* | 143 | $50k to $70k, to $60k |
| *recruiter* | 325 | Resource Spectrum |
| *state* | 462 | TX, Texas, Miami, Georgia, MI |
| *city* | 639 | Austin, Battle Creek, San Antonio |
| *country* | 363 | US, USA, England, UK |
| *language* | 867 | RPG, COBOL, CICS, Java, c, c++, SQL, PowerBuilder |
| *platform* | 705 | AS400, Windows 95, windows, portable systems, PC |
| *application* | 605 | DB2, Oracle, DB2 server, sysbase |
| *area* | 980 | failure analysis, multimedia, TCP/IP, internet |
| *required years experience* | 173 | 2, 2+, Two, 5, 4 |
| *desired years experience* | 45 | 5, 4, 10 |
| *required degree* | 80 | BS, B.S., Bachelor, Bachelor's, BSCS |
| *desired degree* | 21 | Phd, BS, BSCS, Masters, MSCS |
| *post date* | 288 | 30 Aug 1997, 11 Sep 1997 |

Figure 3.5: Details of the Job Postings dataset

man. It is this free text that contains all the interesting information. The author has employed some formatting such as separating sections with newlines but in general there is not much structure to the message. What structure is available varies from message to message and is not consistent across documents.

The Job Postings dataset has been annotated[1] for 17 fields. Figure 3.5 lists the fields in the dataset along with the number of occurrences of each field and examples of each field. The *id* field is a header that is attached to each message. The *title* is the job title for the particular job being advertised. The *company* and *recruiter* fields refer to the company that has the job available and the recruitment agency that is posting the ad. The *state*, *city* and *country* fields refer to where the job is based. The *salary* field refers to the salary offered and can include words as well e.g. 'to $60k'. The *application* field refers to various computer applications that the job requires while the *language* field refers to computer programming lan-

[1]We use a version of the corpus that is annotated using the standard start-send annotation approach rather than the corpus' original template-style annotations.

<purchabr>LIFETIME</purchabr> {<purchcode>LFT</purchcode>} EXCHANGE OFFER ACCEPTED

NEW YORK, APril 1 - <purchaser>LIfetime Corp</purchaser> said <acquired>Retirement Housing Corp</acquired> has <status>accepted</status> its previously announced acquisition offer.

<acqabr>Retirement</acqabr> will operate autonomously as a separate subsidiary.

Reuter

Figure 3.6: An example Reuters Corporate Acquisition article

guages that are required. The *area* and *platform* fields are less well defined. The *area* field refers to more general categorizations of the job type e.g. multimedia. The *platform* field refers to operating systems and general system types. Both of these fields are poorly defined. The fillers that match these fields vary and are inconsistent: fillers are marked as *area* or *platform* in some documents but not in others. The *required years experience* and *desired years experience* fields are very similar and can be difficult to distinguish. Similarly *required degree* and *desired degree* are often similar and difficult to distinguish. Sometimes the *desired degree* is a higher degree than the required one, but IE systems have no concept of the relationship between various degree qualifications. The *post date* is the date that the message was posted and the format is always the same. In this dataset *language*, *platform*, *application*, and *area* are multi-valued fields while the rest are single-valued. When they occur in a document there are often several of them i.e. if the document contains the *language* field, several different *language*s are usually specified. Some fields are much more common in this dataset than others. The *desired degree* and *desired years experience* fields occur only 45 and 21 times respectively in the dataset. In contrast the *area* field occurs 980 times and the *language* field 867. When evaluating the performance on this dataset we should keep in mind how often each field occurs in the dataset.

### 3.2.3 Reuters Corporate Acquisitions

The Reuters Corporate Acquisitions dataset consists of 600 articles taken from the Reuters newswire describing acquisitions of companies.

Figure 3.6 shows an example from this dataset. The language in these articles is not grammatical English. It is generally brief and terse. There is very little

| Field | Occurrences | Examples |
|---|---|---|
| *purchaser* | 624 | Sahlen and Associated Inc |
| *purchabr* | 1263 | Sahlen, Sahlen and Associates |
| *purchcode* | 279 | TIRR, |
| *acquired* | 683 | Norcros Plc |
| *acqabr* | 1450 | Norcros |
| *acqcode* | 214 | OEH, NCRO.L |
| *acqbus* | 264 | building, oil and gas interests |
| *acqloc* | 213 | Southern California |
| *seller* | 267 | CSR Ltd |
| *sellerabr* | 431 | CSR |
| *sellercode* | 136 | CSRA.S |
| *status* | 461 | proposed, approved, agreed in principle |
| *dlramt* | 283 | 542.2 MLN STG, almost a billion dlrs, not disclosed |

Figure 3.7: Details of the Reuters Corporate Acquisitions Dataset

structure in the documents. It is annotated with 13 fields, although not all these fields are used in experiments reported by other IE systems.

Figure 3.7 shows the fields along with the number of occurrences and examples of each field. The *purchaser* field is the full unabbreviated name of the purchasing company. The *purchabr* field refers to any abbreviated referencing of the company's name, i.e. any time the company is mentioned without using its full official name. The *purchcode* is the company's stock-exchange code. Similarly *acquired*, *acqabr* and *acqcode* give this information for the company being acquired and *seller*, *sellerabr* and *sellercode* give this information for the company that is selling. This separation of full name and abbreviated name into different fields is difficult for IE systems to deal with. The fields are closely related, but there is no information in the annotation schema and representation that indicates that there is a relationship between the fields.

The *acqbus* and *acqloc* fields describe the area of business that the acquired company is engaged and where it is located. The *status* field refers to the current status of the acquisition and *dlramt* refers to the amount of the acquisition. All the fields in this dataset are single-valued fields. The abbreviated fields are much more common than the other fields, often occurring several times per document

and in different forms. The code fields are much less common, only occurring in a fraction of the documents.

## 3.3 Evaluation of Information Extraction Systems

### 3.3.1 Basic Evaluation

When evaluating IE algorithms we use precision, recall and f-measure. These are standard measures that are widely used to evaluate information retrieval systems. Precision is defined as

$$precision = \frac{TP}{TP + FP}$$

TP refers to true positives which are fragments that were extracted that should have been extracted. FP is false positives and refers to fragments that were extracted that should not have been extracted. Precision indicates the percentage of all the fragments that we extracted that were correct. Recall is defined as

$$recall = \frac{TP}{TP + FN}$$

FN refers to false negatives which are fragments that should have been extracted but were not. Recall indicates the percentage of all fragments that should have been extracted that were actually extracted. F-measure is the harmonic mean of precision and recall and is defined as

$$f1 = \frac{2.precision.recall}{precision + recall}$$

In IE systems there is a trade-off between precision and recall. Increasing one usually decreases the other. If we had a system that extracted a single fragment and extracted it correctly, we would have perfect precision but recall would be low as we would have missed all the other fragments that we should have extracted. If we had a system that extracted every possible fragment it would have perfect recall but low precision because we would have extracted many fragments that we shouldn't have.

### 3.3.2 Evaluation Method

A truly comprehensive comparison would compare each algorithm on the same dataset, using the same splits, and the exact same scoring system. Unfortunately, a conclusive comparison of the different IE systems is impossible using the current published results. The other systems are evaluated using slightly different methodologies [31].

There are several orthogonal issues regarding evaluation. The first is whether to give credit for partial matches (where an extracted fragment is correct at one end, but the other end is the wrong token). We take the more conservative approach of using exact matching only. Thus if the *speaker* is 'Dr J. Lee' and we extract only 'J. Lee', this would count as both a false positive and a false negative. Thus our evaluations are conservative with respect to counting true positives: we must identify both the start and end exactly.

The second issue is how to count correct extractions and errors. The most conservative approach is to require the system to extract all occurrences of a field (all slot occurrences: ASO) to get full credit. Thus if a document contained a target field which had two occurrences, '2pm' and '2:00', then the system would be required to extract both.

An alternative is a "template" approach (one-slot-occurrence: OSO). In this case it is sufficient to extract either '2pm' or '2:00' as they refer to the same entity. It is assumed that there is one correct answer per slot, and the extraction algorithm's most confident prediction is used as its prediction. OSO evaluation makes sense for fields where we know that they refer to a single value (e.g. the time that a seminar starts).

All the systems we compare to use some form of the template filling (OSO) results although they don't specify exactly how they measure performance. BWI for example, assumes one filler per slot and discards all but the most confident predictions.

### 3.3.3 Methods of Counting Extractions: Discussion

In this section we discuss how to correctly count extractions given a rich enough representation. Unfortunately this correct method is not possible using current

standard representation. Thus we also discuss the practicalities of the available methods of counting extractions.

The annotation methods used in the standard datasets does not allow us to represent entity information about field occurrences in documents (see figure 3.1). The field is the general concept that we are trying to extract, e.g. speaker. An entity is an abstract thing that an extracted text string represents. Different extracted text strings (fillers) can refer to the same entity. For example, 'Professor Sara Kiesler' and 'Prof. Kiesler' both refer to the same entity - a person named 'Professor Sara Kiesler'. Different fillers can refer to the same entity or to different entities. Thus a field can be multi-valued or single valued. Single-valued fields are those where all fillers refer to a single entity. For example, *stime* is a single-valued field as a seminar can only have a single start-time. Different fillers such as 3, 3:00 and 3pm all refer to the same entity. Multi-valued fields are those that can refer to several different entities. For example, a seminar can have multiple speakers. A document with fillers 'Professor Sara Kiesler', 'Prof. Kiesler', 'Woody Vaskula' and 'Vaskula' refers to two entities.

When evaluating an IE algorithm we wish to count how many correct and incorrect extractions the system made. The task in IE is to identify the entities present in documents. Current systems identify fillers rather than entities. It should be sufficient to identify each entity rather than each filler. For example, if we extract that the start-time of a seminar as 3pm but fail to identify 3:00 as a start-time it should not matter. For single-valued fields this means that if we identify one filler per document and that document is correct we should get full credit for that document. For multiple-valued fields we would need to identify one filler for each entity in the document. This seems the most sensible and correct method for evaluating an IE system but it is not possible to use this method correctly with the current annotation schemes.

This gives two possibilities for evaluating with the current annotation scheme. The first scheme assumes that all fields are single-valued. Thus to get full credit for a document you need to identify one filler for each field in the document. We will call this OSO (one slot occurrence) evaluation. This assumption is incorrect for multi-valued fields such as *speaker*. When a document has multiple *speaker* entities we can get full credit for identifying only one of them. On the Job Post-

ings dataset several of the fields can have many different occurrences in a single document.

The second method of evaluation is to assume that all fields are multiple-valued and all fillers refer to different entities. Thus to get full credit for a document we need to identify all fillers for all fields in the document. We will call this ASO (all slot occurrence) evaluation. This assumption is incorrect for single-valued fields and multi-valued fields where any entity has more than one different filler.

Neither of these methods are ideal. We cannot correctly evaluate IE systems because of the limitations of the annotation system used (except in the case where all fields are single valued). OSO evaluation will overestimate the performance of a system on multi-value fields while ASO will underestimate its performance on single-value fields.

Another issue is whether to assign any credit for partial matches. For example, if the *speaker* of a seminar 'Professor Sara Kiesler' and the system extracts the filler 'Sara Kiesler' it is still a useful piece of information although not exactly correct. Some other IE systems assign partial credit for partial matches. A more complex entity annotation scheme might recognize that these two fillers are the same abstract entity.

## 3.4 Experimental Setup and Document Corpora

When comparing against other systems we use their published results for comparison. These systems usually don't specify in detail the methods of evaluation used. There is no consistent evaluation methodology used by other IE systems so it is difficult to compare directly against other systems. Often different systems use different evaluation methodology and don't describe the evaluation methodology used in their publication. To fairly compare our system to other IE systems we use the most conservative evaluation metric when evaluating our own system. Various systems used variations of OSO evaluation and some of them give credit for partial matches. When evaluating our systems, we give no credit for partial matches and we use ASO evaluation because it is more conservative than OSO evaluation. Thus our evaluations are conservative in relation to other IE sys-

tems published results and are likely understated in relation to competitor systems. (Note: The Pascal Challenge standardized on ASO evaluation with no credit for partial matches). The other systems that we compare against generally use less conservative methods than us in our published results. $(LP)^2$ gives credit for partial matches and the originally published results use OSO evaluation. The HMM results of Freitag and McCallum assume there is one correct answer per document so it is a form of OSO evaluation. BWI also uses OSO evaluation. Lavelli *et al. [26, 31, 30]* describe in more detail the evaluation used by other systems.

We used a 50:50 split of the dataset repeated 10 times. Results for BWI, RAPIER and $(LP)^2$ come from other sources [31], and where possible we use the same splits for each system.

For the Pascal Challenge dataset the evaluation methodology was different than for the other IE datasets. The Pascal Challenge evaluation methodology was precisely defined and consisted of a four-fold cross-validation of the training data and a held-out test set. For the cross-validation experiment the training data was divided into four groups. Each group is used for the testing with the other three groups being used for training. The other experiment involved training on all the training data and testing on all the test data. Because it used different methodology and compared different systems, we leave discussion of the Pascal Challenge to a separate chapter.

## 3.5 Summary

In this chapter we discussed some of the issues surrounding the annotation of datasets for IE.

- We described the three standard IE benchmark datasets: the Seminar Announcements, the Job Postings and the Reuters Corporate Acquisitions.
- In the past evaluation of IE systems there has been little standardization of IE evaluation methodology.
- We described what we believe to be the correct method of evaluating IE systems. Unfortunately it is not possible given the current standard annotation

schema.

- We described two methods of evaluation that are practical with current annotation schemes: OSO assumes that all occurrences of a field refer to a single entity and requires the IE algorithm to extract one occurrence of the field per document. ASO is more conservative and requires the IE algorithm to extract all occurrences of a field in a document.

- Our evaluation methodology is conservative with respect to the results presented by other IE systems. Thus, even though it may they may not be directly comparable, our results are at worst understated in relation to the competitors.

# Chapter 4

# A Classification Approach to Information Extraction

## 4.1 Overview

In chapter 2 we gave an overview of Machine Learning and how it can be used for prediction and for Text Classification in particular. In this chapter we will introduce a basic approach to Information Extraction using Machine Learning. This basic approach treats IE as a classification task and combines separate models for identifying the start and end of a fragment with a simple method for deciding which starts to pair with which ends. We will expand and enhance this system in the following chapters as well as investigate the various aspects of the systems performance.

We are using a Machine Learning approach to IE so our system consists of two distinct phases: learning and extracting. We take a supervised approach to learning. In the learning phase our system uses a set of labelled documents to generate models which we can use for future predictions. The extraction phase takes the learned models and applies them to new unlabelled documents using the learned models to generate extractions.

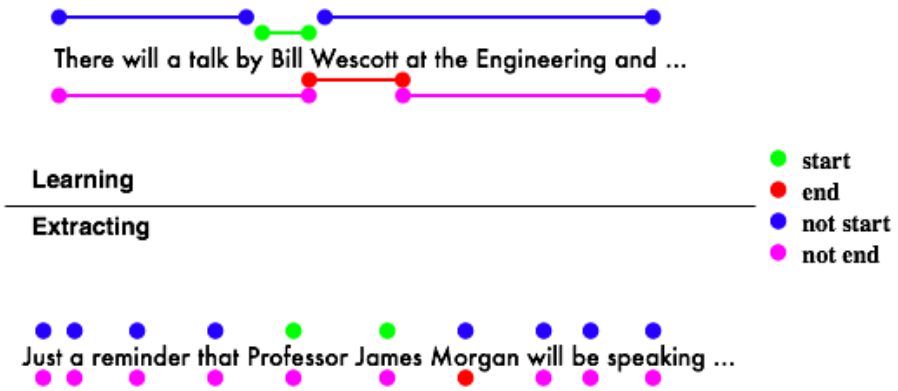


Figure 4.1: Information Extraction as classification

## 4.2 Information Extraction as Classification

In chapter 2 we described the standard approach to Text Classification. IE is a token classification task rather than a Text Classification task. With IE we are working with texts but the basic unit that we are seeking to classify are tokens in the text rather than the entire text. With Text Classification we are seeking to identify whether an entire text is a member of particular category. With IE the categories are start and end, and the objects we seek to assign to these categories are the individual tokens.

With TC we represent entire documents using a binary bag-of-words vector. With IE we are representing individual tokens. We cannot use the BOW approach as used in TC as each token is only a single word. We much encode additional information about the token to enable our learning algorithm to generalize. For IE we encode several features of the token as well as relational information about the surrounding tokens.

The features include the specific token, as well as part-of-speech (POS), chunking, orthographic and gazetteer information. The features are described in more detail in chapter 5 and in appendix C. In addition, we add features to represent a fixed-width window of tokens on either side of the instance's token. The learning algorithm uses these features to create a model that can distinguish between tokens that are starts of fields, ends of fields or neither.

We treat IE as a classification task. Following [20, 12], the approach that we use is to treat the identification of fragment start and end positions as distinct token classification tasks. The instances are all tokens in the document. All tokens that begin a labelled field are positive instances for the start classifier, while all the other tokens become negative instances for this classifier. Similarly, the positive examples for the end classifier are the last tokens of each labelled field, and the other instances are negative examples.

Figure 4.1 gives an example of what we mean by IE as classification. There are two classifiers - one to identify starts of fragments and another to identify ends of fragments. Each token is classified as being a start or non-start and an end or non-end. When we classify a token as a start, and also classify one of the closely following tokens as an end we extract the fragment between these two tokens.

The system consists of two different phases - learning and extracting. In the learning phase, the system uses annotated documents to learn to identify starts and ends of the field we wish to extract. Each token in the document is represented by a single instance. Each instance has a set of features that describe the given token.

In the example given in Figure 4.1 the token 'Bill' is positive example for the start-of-field model and all the other tokens are negative examples. The token 'Wescott' is a positive example for the end-of-field model and all other tokens are examples of tokens that are not the end of a field. The system uses these two sets of examples to learn two different models - one that can recognize the start of a field and one that can recognize the end of a field.

One we have learned these models from our training data, we can use the learned models to extract fillers from new documents. In the extraction phase we apply the two learned models to each token in the document. We mark each token as being start/not-start and end/not-end. Thus each token can be a start, end, neither or both. After this phase we match up starts and ends that were predicted by our model. In the example shown we have predicted one end and two starts. We must decide which (if any) of the starts to match with the end to form an extracted fragment.

The extraction of fillers follows the token classification phase. We use a simple histogram model. In the example above there are two possible extracted fillers: 'James Morgan' and 'Professor James Morgan'. We can extract both fillers or we

can extract the fragment that has the highest confidence. We estimate confidence as $C_s * C_e * P(|e - s|)$. $C_s$ is the confidence for the start prediction. $C_e$ is the confidence for the end prediction. We estimate the confidence in the start and end predictions for an instance as the distance of that instance from the hyperplane relative to the maximum distance seen in the training data. $P(|e - s|)$ is the probability of a fragment of that length which we get from the tag-matcher histogram.

To summarize, this IE classification approach simply learns to detect the start and end of fragments to be extracted. It treats IE as a standard classification task, augmented with a simple mechanism to attach predicted start and end tags. During the training phase we record the length of each field occurrence. From this length histogram we calculate the probability of a filler of that length occurring. We use this probability to decide whether or not to extract potential fillers. When we identify starts and ends in close proximity to each other with the end following the start, the probability of extracting the filler is estimated from the length histogram. Our experiments demonstrate that this approach generally has high precision but low recall. We will refer to this simple IE as classification approach as ELIE$_{L1}$ (or L1).

Fig 4.2 summarizes the learning process. The set of training examples are converted to a set of instances for the start and end tags as described above. We will describe the instance representation in more detail in the next chapter. Each token in each training document becomes a single instance, and is either a positive or negative example of a start or end tag. Each of these instances is encoded according to several features for the particular token in question and the tokens surrounding it. Then the attributes are filtered according to Information Gain (This process is described in chapter 5). These instances are passed to a learning algorithm which uses them to learn a model. At the end of the L1 training phase we have models for start and end tags and a length histogram for the start-end pairs.

The start-end pairs are passed to the tag-matcher which is charged with matching start and end tags. Our experiments involve a tag-matcher which generates a histogram based on the number of tokens between each start and end tag in the training data. When matching predictions, the probability of a start-tag being paired with an end-tag is estimated as the proportion with which a field of that

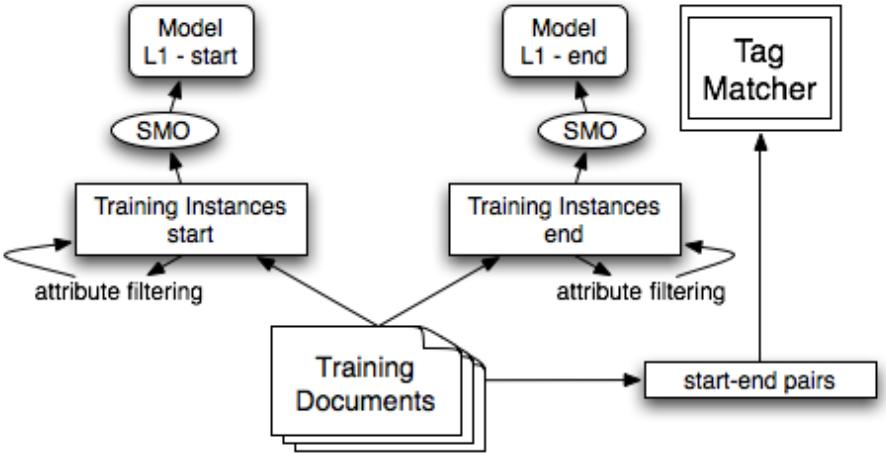


Figure 4.2: L1 Learning

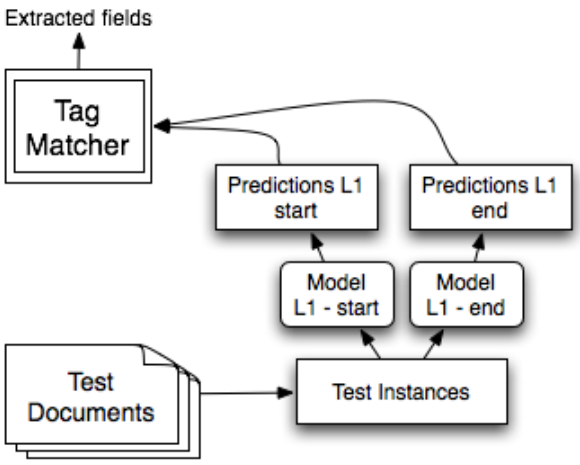


Figure 4.3: L1 Extracting

length occurred in the training data. This approach performs adequately and we don't focus on the tag-matching further.

Once we have learned the models from the training data, we can apply them to new documents in order to extract fragments from them. Figure 4.3 summarizes the extraction process. The documents we wish to extract from are converted into a set of test instances that have the same representation as the training instances.

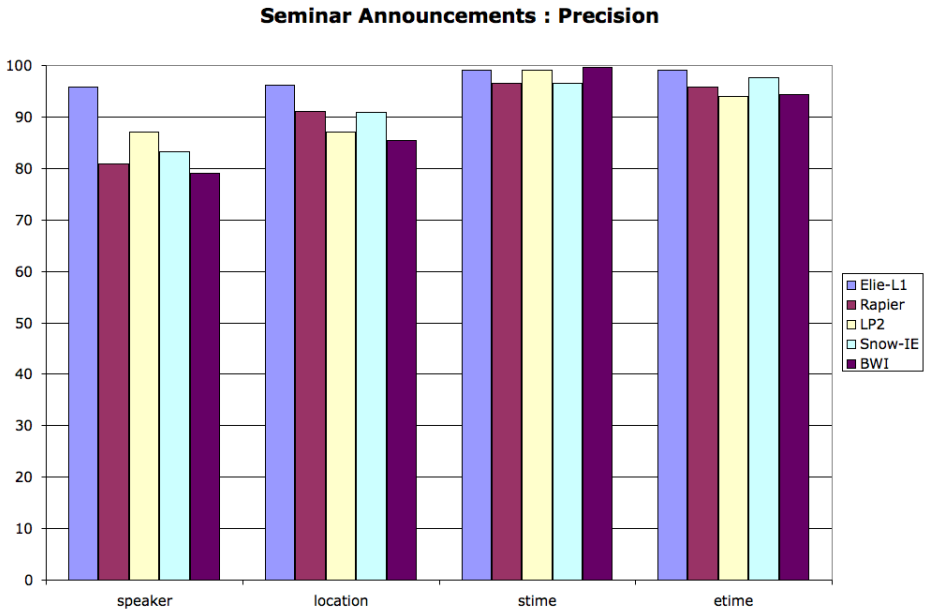


Figure 4.4: L1 Precision for the Seminar Announcements dataset

We then apply the models for start and end to these instances. This gives us a set of predictions for a set of starts and ends for the field that we want to extract. These predictions are passed to the tag-matcher. The tag-matcher uses the probability information from the training phase to decide which starts to match to which ends. Matching the predicted starts to predicted ends results in a set of extracted fragments for the field we wish to extract.

## 4.3 Evaluation

We evaluate this simple learner using the methodology described in chapter 3. We evaluated it on the three standard datasets and compared it to the other systems described in chapter 2. There are several parameters that can be varied for this IE system. We examine the effect of varying some of these parameters in the next chapter. For the experiments presented here we use a default set of parameters.

Figure 4.6 shows the performance of this basic IE as classification approach on the seminar announcements dataset. The results are presented in three graphs

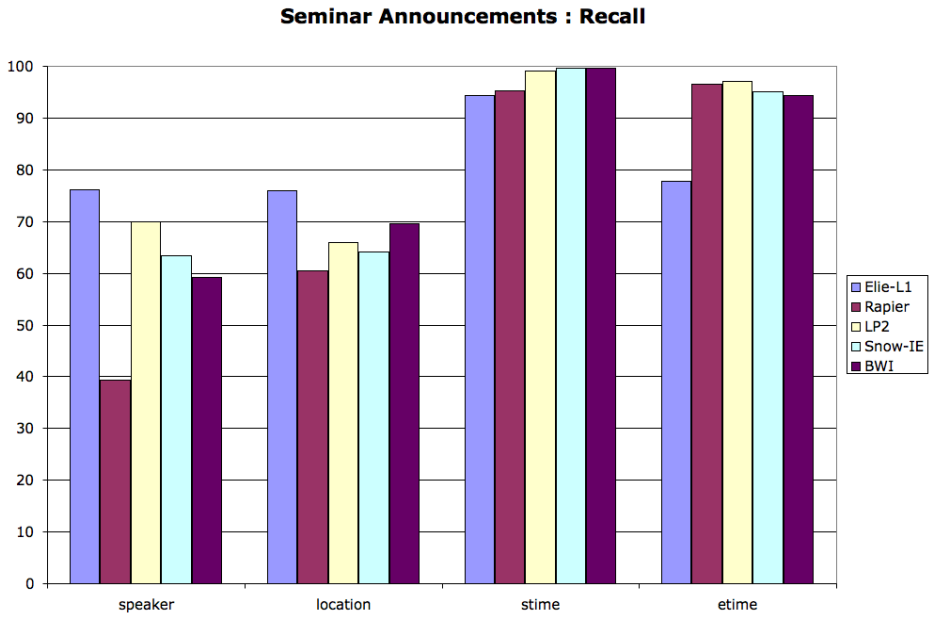


Figure 4.5: L1 Recall for the Seminar Announcements dataset

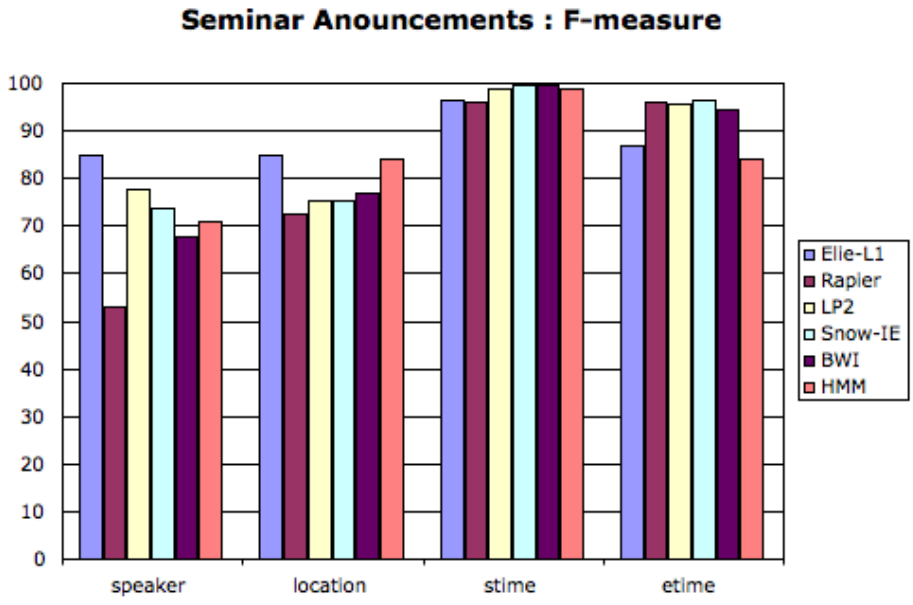


Figure 4.6: L1 F-measure for the Seminar Announcements dataset

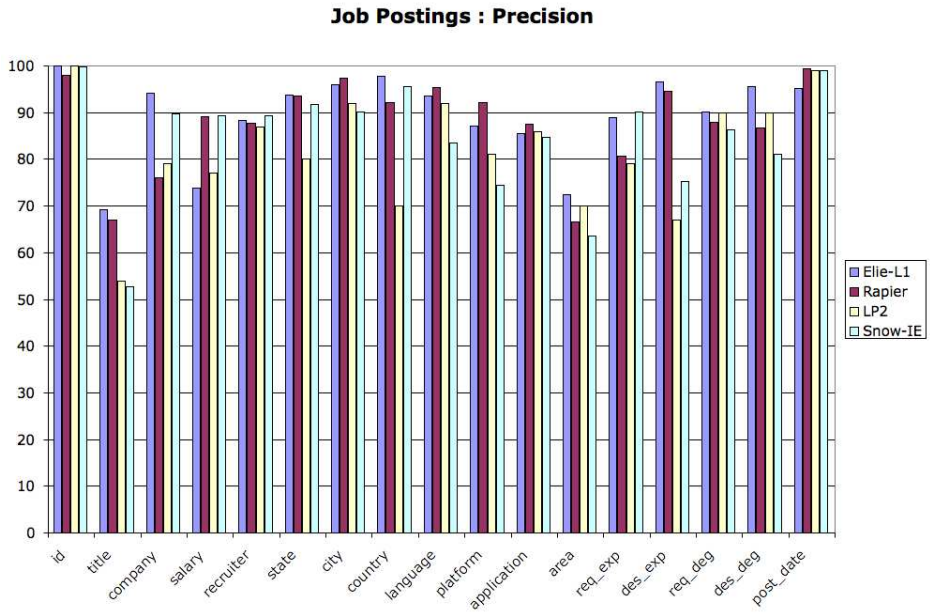


Figure 4.7: L1 Precision for the Job Postings dataset

showing precision, recall and f-measure (Note: for HMM we only have f-measure results). On this dataset, the L1 approach generally outperforms the other IE systems.

Our approach has the highest precision of all five IE systems on three of the four fields (*speaker, location, etime*) second best precision on the other field. ELIE$_{L1}$ has the best recall on two of the fields (*speaker* and *location*) while it has the worst recall on the *etime* field. When we consider f-measure, our system has the highest f-measure on the *speaker* and *location* fields while it has the worst f-measure on the *etime* field. The *speaker* and *location* fields are generally regarded as the more difficult fields in this dataset while the *stime* and *etime* are regarded as being easier and more structured.

Figure 4.7 shows performance of ELIE$_{L1}$ on the Job Postings dataset compared to Rapier, $(LP)^2$ and SNoW-IE. Again we show precision, recall and f-measure. In total there are 17 fields in this dataset. When we consider precision, ELIE$_{L1}$ performs best of all on nine of the 17 fields. It has the worst precision for 4 of the 17 fields. When we look at recall, ELIE$_{L1}$ performs best on only one of the fields and is the worst performing system on 2 of the fields.

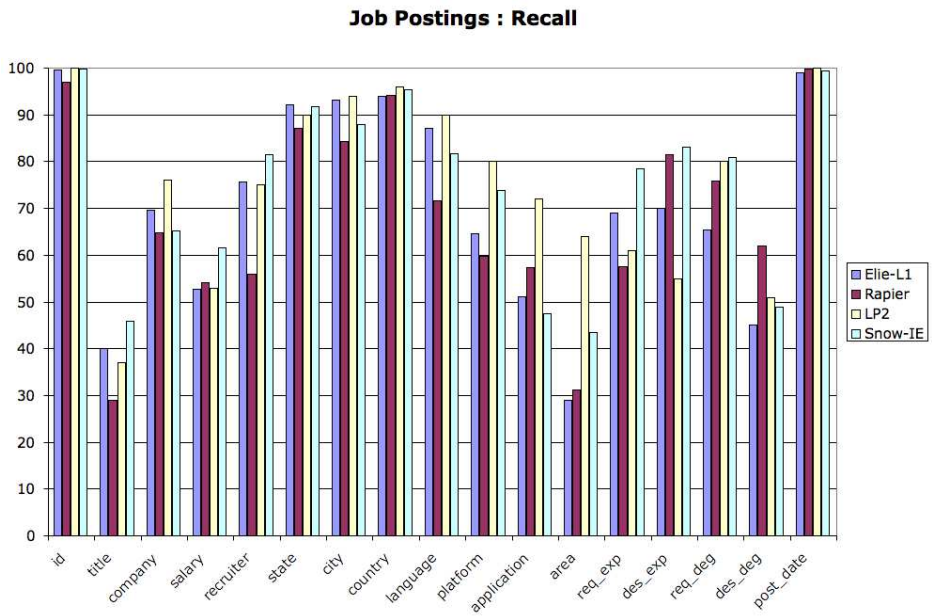


Figure 4.8: L1 Recall for the Job Postings dataset

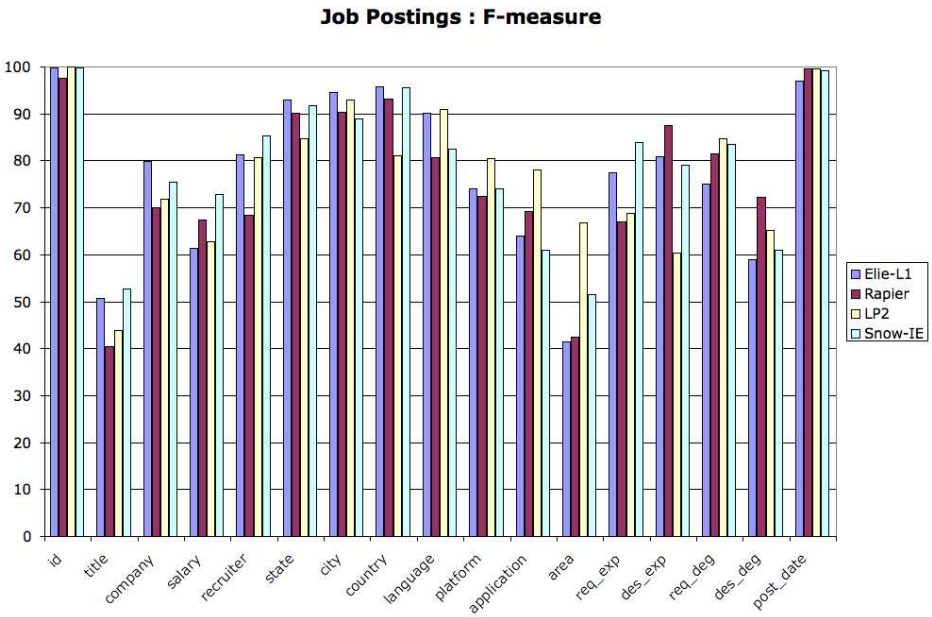


Figure 4.9: L1 F-measure for the Job Postings dataset

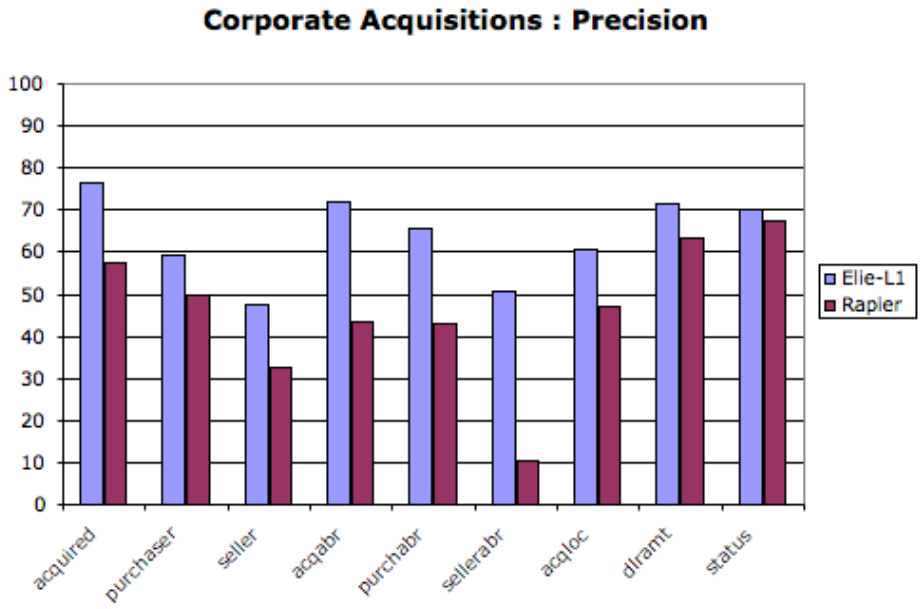


Figure 4.10: L1 Precision for the Reuters Corporate Acquisitions dataset

For f-measure, ELIE$_{L1}$ is best for 5 of the fields and it is the worst performing system for 2 of the fields.

We conclude that the approach is competitive with the other systems on this dataset. For most fields it is one of the better performing systems and it is rarely the worst performing system.

Figures 4.10, 4.11 and 4.12 shows the performance of ELIE$_{L1}$ on the Reuters Corporate Acquisitions dataset compared to Rapier and HMM (f-measure only). ELIE$_{L1}$ has higher precision than RAPIER on every single field. It generally has poorer recall than Rapier, beating it on only 2 fields. HMM has the best f-measure on each of the fields that we have results for.

We conclude that ELIE$_{L1}$ is competitive with Rapier on this dataset. It has higher precision on all the fields. Its f-measure is sometimes better and sometimes worse than Rapier. HMM performs best on this dataset.

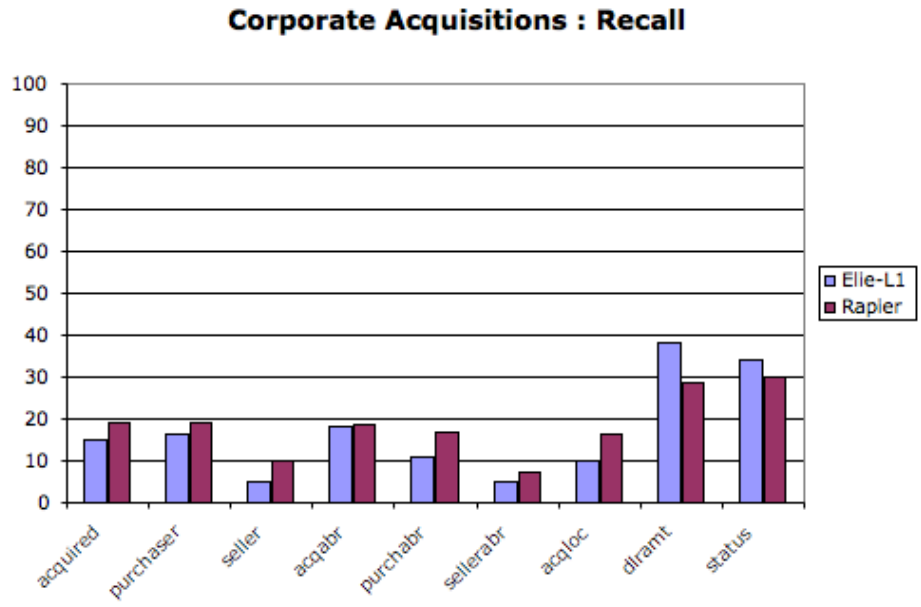


Figure 4.11: L1 Recall for the Reuters Corporate Acquisitions dataset

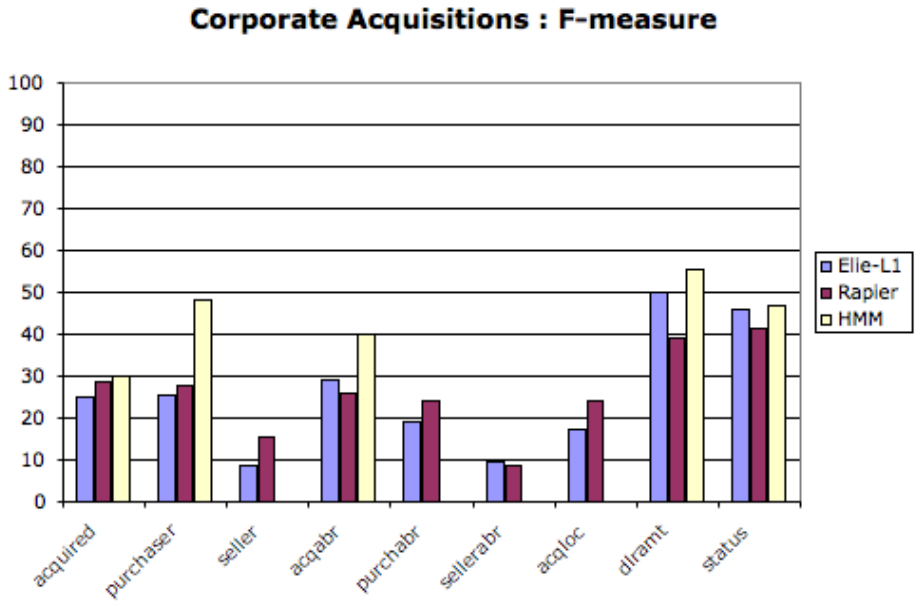


Figure 4.12: L1 F-measure for the Reuters Corporate Acquisitions dataset

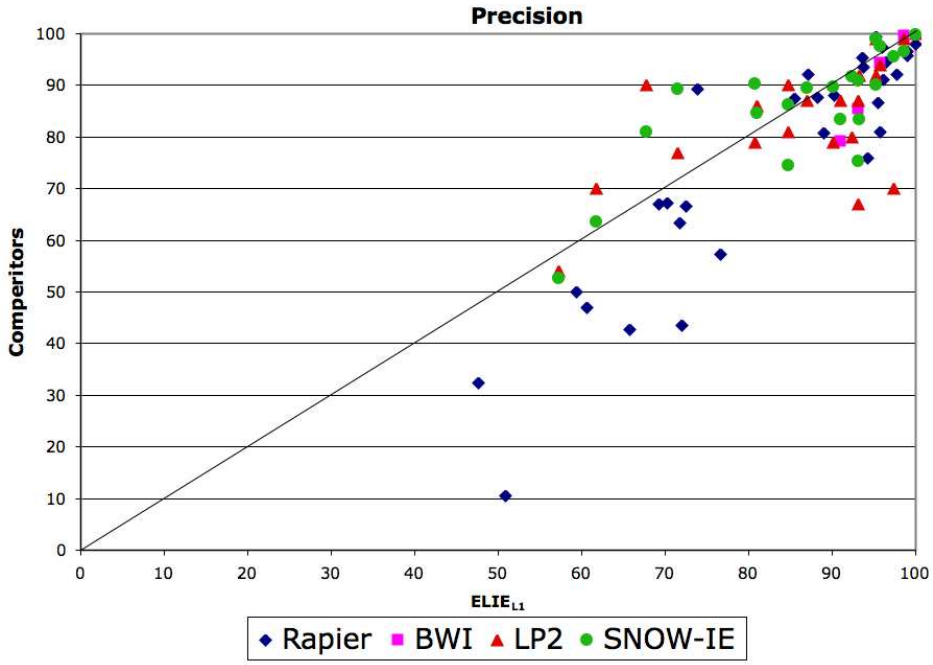


Figure 4.13: L1 Precision summary

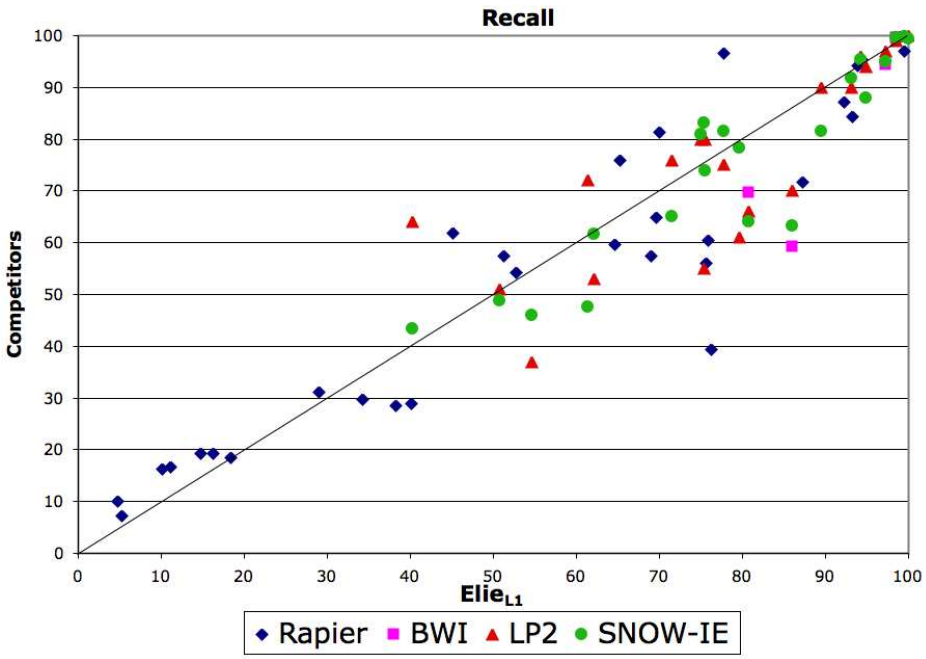


Figure 4.14: L1 Recall summary

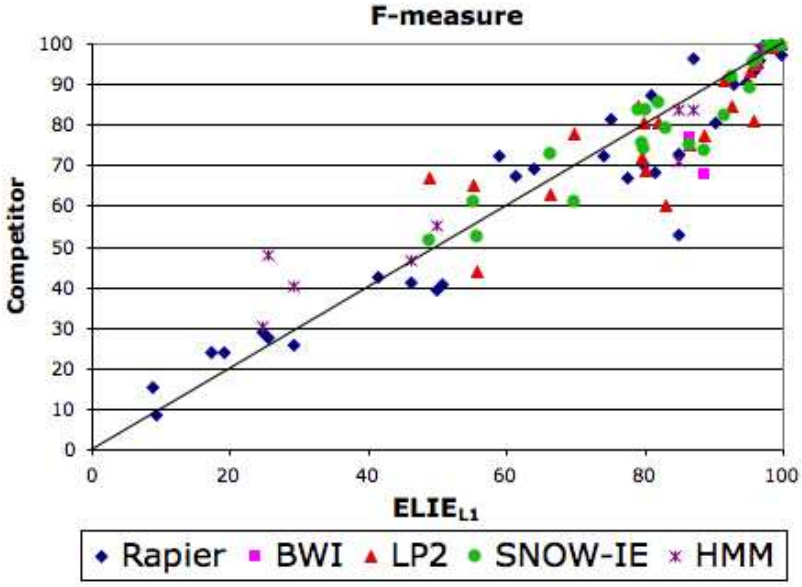


Figure 4.15: L1 F-measure summary

## 4.4 Discussion

Figure 4.13, Figure 4.14 and Figure 4.15 summarize the performance of ELIE$_{L1}$ against that of other IE systems. The horizontal axis shows the performance of ELIE$_{L1}$ while the vertical axis shows the performance of the competitor system. Each point represents the performance of ELIE$_{L1}$ vs a competitor on a single field. Points that occur above the diagonal line indicate that the competitor system is doing better while points occurring below the diagonal line indicate that ELIE$_{L1}$ is doing better. On the precision graph, most of the points are below the diagonal. For recall, the majority of points are below the diagonal or clustered close to it. For f-measure, most of the points are below the diagonal. Those that are above it are generally close to it.

We conclude that the approach described is competitive with the other IE systems. In fact on many fields, this simple approach outperforms the state-of-the art competitor IE systems.

In the following chapters we investigate variations of the 1-level classification approach and techniques for augmenting it. We investigate which aspects of the

algorithm contribute to its performance. In chapter 5 we discuss the feature representation used in more detail and investigate how much the features contribute to performance. We also investigate the overall effect on performance of attribute filtering and the choice of learning algorithm. In chapter 6 we discuss the effect of dataset imbalance and instance filtering on the performance of the system.

Then in chapter 7 we extend ELIE$_{L1}$'s one-level classification approach and introduce a two-level classification approach that substantially improves performance.

## 4.5 Summary

In this chapter we described a simple 1-level classification approach to IE (ELIE$_{L1}$). This approach represents IE as a token classification task by combining separate SVM classifiers for identifying starts and ends of fields and matches up starts and ends using a simple histogram model.

We evaluated this approach using three standard IE datasets, compared it to several other state-of-the art IE algorithms and showed that it is competitive with other state-of-the-art IE systems. This approach was the best performing system on the SA dataset, it was competitive with the other IE algorithms on the Job Postings dataset and was competitive with Rapier while being outperformed by HMM on the Reuters dataset. It gave high precision compared with other systems whereas recall was not as competitive.

# Chapter 5

# Features, Attributes and Learning Algorithms

In order to apply Machine Learning techniques to any classification task we must supply a set of examples for the training algorithm. The learning algorithm then uses these examples as a basis for building a model with which future predictions can be made. Each example instance is described by a set of features that describe various properties of the instance. We use a vector space model for our features. The features are not defined in advance but depend on the tokens that occur in the documents.

In the case of IE each of these examples is a token. Each token is an example of a start, end or neither. Since each example is just a single token, the learning process requires that a set of features be identified and associated with each token. These features are used to represent the token and are used by the learning algorithm to differentiate between tokens.

Each instance is characterized by a set of attributes. Each attribute measures a different aspect of the particular instance. In the example of Machine Learning that we gave in Figure 2.1 the attributes were pre-defined and fixed in advance. However when dealing with text, it is not possible to fix the attributes in advance as the attributes depend on the tokens that occur in the text. The choice of features to be used as attributes can affect the quality of the model generated by the learning algorithm. It is important to identify the salient features. We want to identify

features that generalize well and are not too specific as these features will be useful in classifying instances that are different to those seen in the training data.

## 5.1 Features and Encoding

In chapter 2 we discussed the representation of text for Machine Learning. Our approach to representing text documents for IE is similar but adapted for the IE task.

Documents are broken into tokens. Each token is a single instance. Each instance can be a positive or negative example for the start or end of the field we are trying to extract. A token is defined as any continuous sequence of alphabetic or numeric characters. Punctuation symbols are treated as separate tokens. Each instance has several kinds of features associated with it. These feature-types are:

**Token** The actual token.

**POS** The part-of-speech of the token. Each token is tagged with its corresponding POS using Brill's POS tagger [6]. We also represent chunking information about the tokens. The POS tags are also grouped into noun-phrases and verb-phrases (Chunk).

**GAZ** The values associated with the token in a gazetteer. The gazetteer is a user-defined dictionary. It contains lists of first-names and last-names taken from the U.S. census bureau, a list of countries and cities, time identifiers (am, pm), titles (Jr., Mr), and a list of location identifiers used by the U.S. postal service (street, boulevard). Pre-defined sequences of features that match pre-defined entities are also marked as entities (ERC). The entities recognized are 'person' and 'time'. For example the sequence of gazetteer features 'firstname' followed by 'lastname' would additionally be marked as 'person'.

**Orthographic** These features give various orthographic information about the token. Examples of these features include whether the token is upper-case, lower-case, capitalized, alphabetic, numeric or punctuation.

**Pair** In addition to representing all the above features separately, we also add features are all possible pairs of the POS, GAZ and orthographic features. For example, a token that is a name and is capitalized could be represented by a single feature.

All features are binary. Every instance is represented as a vector of all the features. Each feature can have value 1, indicating the presence of the feature in that instance, or 0 indication the absence of the feature in that instance.

There are two kinds of features that are abstractions of the basic feature types listed above. Chunk features are an abstraction of the POS features. ERC features are a higher level abstraction of the GAZ features. The idea behind the Chunk and ERC features is to reduce the burden on the learning algorithm, by encoding some potentially useful information. For example, the learner should be capable of learning that a name is often a first-name followed by a last-name, but if we add a feature to encode this information it should allow the learner to focus on whether the name should be extracted or not.

To represent an instance, we encode all these features for that particular token. In addition, for a fixed window size of w, we add the same features for the previous w tokens and the next w tokens. For example, the string

```
Place:     WeH 4601
Speaker:   Alex Pentland
```

would be tokenized and tagged as

| Token | POS | Chunk | Gaz | ERC | Orthographic |
|---|---|---|---|---|---|
| place | NNP | | | | alpha, cap |
| : | | | | | punct |
| weh | NNP | | | | alpha, cap |
| 4601 | | | | | num |
| \n | | | | | |
| speaker | NNP | | | | alpha, cap |
| : | | | | | punct |
| alex | NNP | NPs | firstname | person | alpha, cap |
| pentland | NNP | NPe | lastname | person | alpha, cap |

If we encode the instance centered at the token *alex*, using a window of size 1 to encode relational information about the next and previous tokens, we would get the following features:

```
Tok_alex_0, Tok_:-1, Tok_pentland_+1,
POS_NNP_0, POS_NNP_+1,
C_NP_s_0, C_NP_e_+1,
E_person_0, E_person_+1,
O_aplha_0, O_cap_0 , O_punct_-1,
O_aplha_+1, O_cap_+1
```

All tokens in the dataset become a single instance and are encoded in this way. Each instance is represented as a binary vector of all the attributes that occur in the training set. Each instance is encoded using this vector with the presence or absence of a particular feature being represented by a 1 or 0 at the vector position that represents that feature. There are a large number of attributes and the vectors are very sparse. For example training on 50% of the Seminar Announcements dataset gives approximately 74K attributes. Training on 50% of the Job Postings dataset gives approximately 43K attributes, training on 50% of the Acquisitions dataset gives approximately 48K attributes while a train-split of the Pascal CFP training corpus gives approximately 170K attributes.

Because of the large number of attributes we filter the attributes according to Information Gain (see section 5.2). It is also possible to filter the negative instances at this point in order to reduce learning time and alter the prior probabilities of the learner (see chapter 6).

In addition to the features described we create new features using pairs of features. The combination of several features could be much stronger evidence of the class of an instance than the presence of the features individually. While the learning algorithm should give higher weight to instances with multiple informative features, adding pair features may help the learning algorithm to recognize these. For example, if an instance is a *firstname* and the next token is a *lastname*, than adding a feature '*firstname* and next *lastname*' may help the learner's ability to generalize.

We create features for all possible pairs of non-token features: i.e. we create a feature for all possible pairs of the features that occur in the POS, orthographic and gaz features. We exclude the token features because there are a large number of token features. Representing all possible feature pairs including the token features would result in a huge increase in the number of features.

## 5.2 Attribute Filtering

Our method of representing instances generates a large number of attributes. This is because we use the tokens as features. Excluding the token features gives a small fixed number of features. For example, using 50% of the Seminar Announcements for learning with a window of 4 gives a representation with over 70,000 attributes. Many of these attributes are rare and occur in only a few instances. Many of them occur too often across both classes to be useful. The vast majority of features have no discriminative value.

Having such a large feature-set results in large learning times for the learning algorithm. We filter the majority of the attributes to reduce learning time. We only wish to use features for learning that are useful for discriminating between classes.

We filter attributes according to Information Gain [40]. Information Gain estimates the amount of extra information that we get from having the attribute present. It gives us an estimate of how well a particular attribute separates the training instances according to their correct classification. We only want to use attributes that give us some new information that we can use to discriminate between classes.

Entropy measures the amount of disorder in a system. In the case of Machine Learning, it measures homogeneity of the instances. For a set of instances S for a binary classification task, we can measure the entropy of S as

$$Entropy(S) = -p^{+}log_{2}p^{+} - p^{-}log_{2}p^{-}$$

where $p^{+}$ is the proportion of positive instances in S and $p^{-}$ is the proportion of negative instances in S.

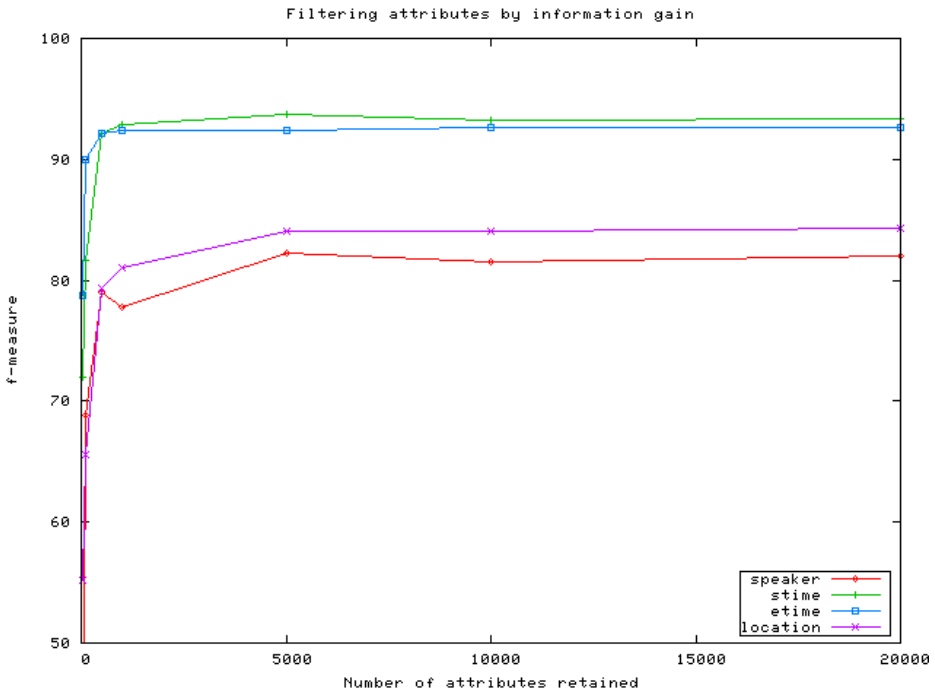


Figure 5.1: Filtering attributes by Information Gain

Using an attribute that is highly discriminatory for the target class will give a large reduction in entropy while attributes that have no discriminatory value will have no effect on the entropy.

The Information Gain of an attribute is defined as the expected reduction in entropy that occurs when we split the instances according to that attribute. The Information Gain of an attribute A, relative to a set of examples S is defined as

$$Gain(S, A) = Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

*Values(A)* is the set of all possible values for attribute *A*. $S_v$ is the subset of *S* for which attribute *A* has value *v*. *Gain(S,A)* gives us the expected reduction in entropy caused by knowing the value of attribute *A*.

Figure 5.1 shows the performance on the Seminar Announcements dataset with various levels of attribute filtering. The vertical axis shows f-measure while the horizontal axis shows the number of attributes used for learning. For each

of the four fields, there is no benefit in using more than 5000 attributes. This indicates that most of the attributes that are generated are not useful for learning. In fact the system doesn't require a large number of attributes to perform well. For the *stime* and *etime* fields, the performance with 1000 attributes is not significantly worse than with 5000 attributes. For the other two fields, *speaker* and *location*, the performance continues to increase up to 5000 attributes. If we use more than 5000 attributes, the performance does not increase any further. However it does not decrease either. This indicates that the extra attributes do not improve our ability to learn but neither do they harm it, e.g. by introducing noise. This indicates that filtering itself does not improve accuracy. If it did, we would expect to see it drop as less attributes are filtered. The only thing the extra attributes contribute is extra learning time. From this we conclude that only a small portion of all the attributes are necessary for learning. Removing most of them will improve the execution time of the algorithm but will not affect the accuracy. We conclude that it is sufficient to use the top 5000 attributes as ranked by Information Gain for learning our models.

We rank all attributes according to Information Gain and pick the top $n$ attributes for representation. In experiments we set $n$ to 5000. This is ELIE's default value for all experiments.

## 5.3 Features-sets and Performance

Figure 5.2 show the performance of the various kinds of features on the Seminar Announcements dataset. The vertical axis gives f-measure while the horizontal axis shows the various fields. We show the performance for the system using only token features (Tok), token features with either POS (Tok + POS), orthographic (Tok + Orth) or gazetteer features(Tok + Gaz), all features except the pair features (All - Pair), all features except token features (All-Tok) and all features (All).

Adding the pair features gives no improvement in performance. On the *stime* and *etime* fields the performance using only the token features is just as good as performance with any of the more general features added. This indicates that these fields are specific and limited in the tokens that occur in them and that there is little scope for improvement using generalization with the more specific features. For

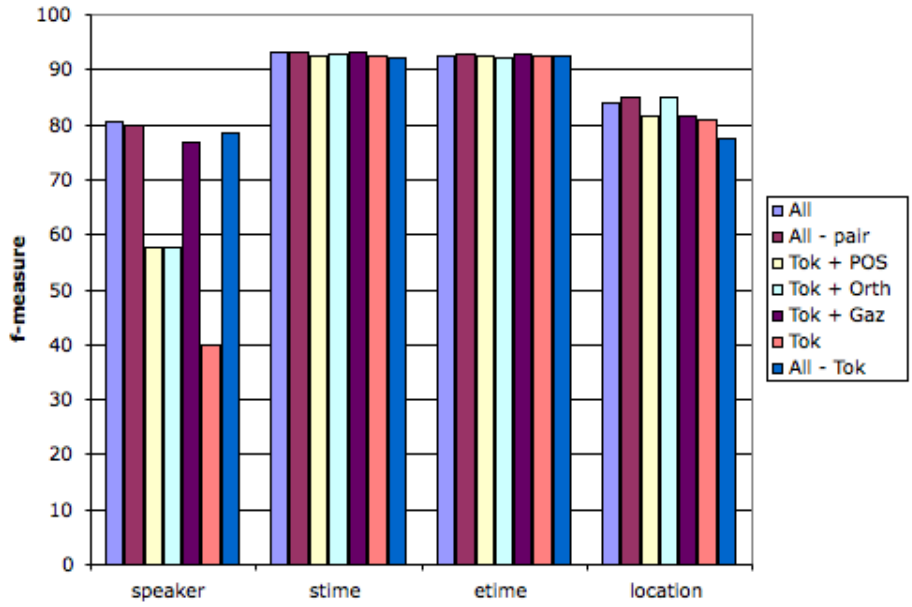


Figure 5.2: The effect of various feature-sets on performance

the *location* and *speaker* fields there is an improvement in performance when the other features are used. This indicates that these fields have more different values and generalization using features such as the gazetteer and orthographic features helps with performance.

For the *location* field adding the orthographic features improves the performance over using the token features alone. For the *speaker* field, there is a large performance improvement using the extra features over using the token features. This is because there is large variability in the *speaker*s in the dataset whereas *stime* and *etime* and to a lesser degree *location* have a more limited number of different values. Thus for the *speaker* field the ability to generalize with features such as whether the token is capitalized or whether it is a proper noun are important for performance. Both the orthographic and the POS feature offer significant improvement over using the token features alone for the speaker field.

Using the gazetteer features offers a large improvement while using all these features further improves performance. Using all features except for token feature gives good performance. It's performance is a good as using token features for

three of the four fields and substantially better for the speaker field. These two feature-sets both give good performance and use different features. These two feature-sets offer two redundant views of the data and they may be suitable for an active learning approach using co-training [5] and multiple redundant views [36].

There is little benefit in adding pair features so ELIE's default feature-set is (All-Pair).

## 5.4 Learning Algorithms

The design of our system is modular. It uses the Weka Machine Learning library [46] for learning the models. We used the SMO algorithm as our learning algorithm for most of the experiments. We can however use any of the Machine Learning algorithms that are implemented in Weka in its place. In this section we investigate some of the other learning algorithms available. Other systems use different learning algorithms - often some form of inductive rule learning. The choice of learning algorithm can have a considerable effect on the performance of an IE system. Other approaches usually compare their systems as a whole and don't consider the effects of the various parts of the system. For example, would $(LP)^2$ perform significantly better if it had a stronger learning algorithm?

Figure 5.3 shows the performance of our one-level classification approach with several different learning algorithms on the Seminar Announcements dataset. The vertical axis shows f-measure. Naive Bayes performs very poorly on this task, being significantly worse than the other algorithms (apart from OneR) on every field. Winnow also performs poorly. It is significantly worse than SMO and Ripper on three of the 4 fields. Ripper performs poorly on the etime field. On the location field it is better than Winnow and Naive Bayes but significantly worse than SMO. On the *speaker* and *stime* fields it is competitive with SMO, almost exactly matching it. ID3 is the second best algorithm on 3 of the 4 fields, outperforming Ripper on three of the four fields. SMO is the best performing algorithm. It performs best on all four fields, with only Ripper able to match it on two of the fields. OneR generally does badly, but does surprisingly well on the *stime* field[1].

[1]In this case OneR generated the following rules for start and end: TOKEN_time_-2 -> start and G_ampm_0 -> end. This means that a token is tagged as start of an *stime* if the token time

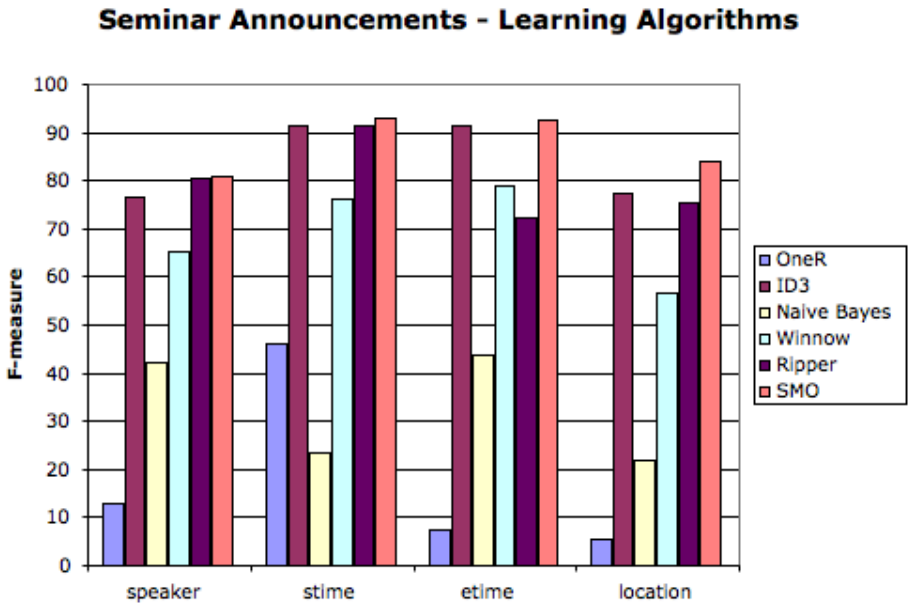


Figure 5.3: Performance of different learning algorithms

This indicates that there is a wide variation in performance depending on which learning algorithm is used. It also indicates that this is a complex task. Naive Bayes often performs very well on text classification tasks but on this task its performance is very poor. The different between the best and worst performing algorithm on each field is large. We conclude that SMO is a good choice of learning algorithm for this task and the choice of learning algorithm can play a large part in the performance of the system. The performance of Ripper indicates that an inductive rule learner can do well on this task.

In summary SMO is the best learning algorithm, Ripper and ID3 perform reasonably well on this task, while OneR, Winnow and Naive Bayes have poor performance. SMO is ELIE's default learning algorithm.

occurs two tokens back and a token is tagged as an end of *stime* if the gazetteer marks it as type 'ampm'.

## 5.5 Summary

In this chapter we investigated some of the various aspects of our IE system that contribute to its performance. Rather than having a monolithic IE system to compare to others, it is important to understand which parts of the IE system contribute to performance. Each different IE system has various caveats that give it an advantage over other systems. But they are usually compared in their entirety. If we separate IE systems into their components and understand which components improve performance we could build an IE system that incorporated the best components of each individual IE system.

We conclude that:

- Token features alone give good performance.
- Other more general features also give good performance and adding them with the token features gives a performance boost.
- The token features and other features form two redundant views of the data which might be suitable for an active learning approach to IE.
- Filtering a large number of attributes doesn't affect performance as long as the number of attributes remains above a certain minimum threshold.
- The choice of learning algorithm is important. Some learning algorithms exhibit poor performance on this task. SMO is a good choice for learning algorithm.

# Chapter 6

# Instance Filtering

## 6.1 Imbalanced Data

Imbalanced datasets are those where the number of examples of one class far outnumbers the number of examples of another class, i.e. the number of negative instances is far greater than the number of positive instances or vice versa. When there are a small number of positive examples it becomes more difficult to learn to identify the positive instances. In many cases however it is the positive instances that are of interest so it is important to identify them. Suppose we wish to automatically identify credit-card fraud and for every fraudulent transaction there are 999 normal transactions. A simple classifier that always predicts that a transaction is OK would be correct 99.9% of the time but it would never identify a fraudulent transaction. Such a system would also have high precision, recall and f-measure. Even though its accuracy is very high[1], it completely fails in its assigned task.

SVMs are one of the best performing learning algorithms on many learning tasks. However their performance drops when used on imbalanced data [3]. Learning with imbalanced data is a problem for all learning algorithms. SVMs perform well on moderately imbalanced data and are not as badly affected by data imbalance as other algorithms. This is because SVM learns from instances that are close to the boundary between classes, i.e. the support vectors. It is not affected by negative instances far from the boundary even if there are many of them.

[1] Assuming accuracy is calculated with respect to both the positive and negative classes.

However, as imbalance increases SVMs performance will suffer.

There are two main approaches to dealing with imbalance in SVMs. The first is apply weighting to the positive examples so that the learning algorithm pays more attention to them and to penalize it more for misclassifying positive examples than negative examples. For SVMs this can be accomplished by using penalty constants for the different classes of data such that errors on the positive instances are more costly than errors on the negative instances.

The second approach is to pre-process the dataset to make it more balanced, either by undersampling the negative examples or oversampling the positive examples. The aim in each case it to make the dataset more balanced. This can be done by removing some of the negative instances (undersampling) or by adding some more positive instances (oversampling). SMOTE [9] performs oversampling by adding new synthetic positive instances. It does this by assuming the regions surrounding positive instances in the instance space and between positive instances are positive and adding positive instances into these areas. It is not clear if undersampling is suitable for SVMs. Removing instances far from the hyperplane should have no effect while removing instances close to the hyperplane could adversely affect performance.

Akbani *et al.* [3] discuss these approaches to dealing with data imbalance and combine the two methods. They describe the causes of performance loss with imbalanced data as being:

1. Positive points lying further from the ideal boundary. With imbalanced data, SVMs tend to learn a boundary that is too close and skewed towards the positive instances (see figure 6.1). Because there are many more negative instances than positive instances, the unpopulated area of the instance space between the positive and negative instances is more likely to contain positive instances than negative instances. If it contained negative instances we are likely to have already seen them. However the SVM places the hyperplane between the positives and negatives so as to maximize margin and minimize error. Thus it is likely that the boundary is too close to the positive instances.

2. Weakness of soft margins. The margin is the distance between the hyperplane and the instances. SVM places the hyperplane so that it is as far as

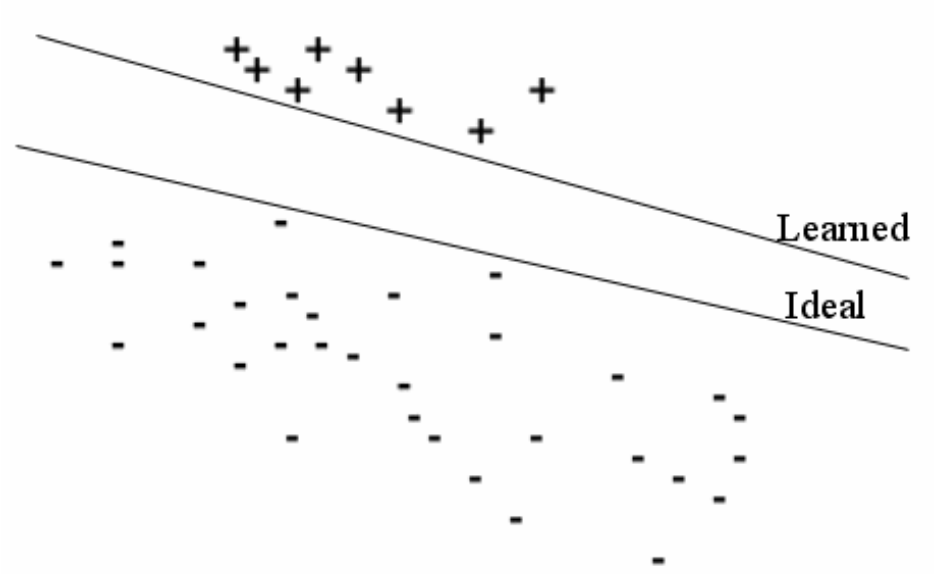


Figure 6.1: SVM and imbalanced data

possible from both the positive and negative instances. Soft margins allow a hyperplane that doesn't separate all the instances, but separates most of them. There is a trade-off between maximizing the margin and minimizing the error. With highly imbalanced data, maximizing the margin would lead us to classify everything as negative. This explains why SVMs fail when the data imbalance becomes very large.

3. Imbalanced support vector ratio. Since the ratio of positive to negative support vectors is imbalanced, the neighbourhood of an instance close to the hyperplane is likely to be dominated by negative support vectors. This means the SVM is more likely to classify an instance that is close to the boundary as negative.

Their experiments show that undersampling improves the distance of the learned hyperplane but the orientation suffers. They claim that undersampling will move the learned boundary closer to the ideal boundary, but may adversely alter the angle of the hyperplane. Despite this claim, their experiments show that undersampling performs better than oversampling with SMOTE and biasing the learner

towards the positive examples. The only approach that performed better than undersampling was their algorithm that combined oversampling and biasing.

Another reason to remove instances is to reduce learning time. Since the SVM only uses instances that are part of the support vectors for learning we can eliminate some of the instances without affecting the accuracy of the classifier. The learning time for a classifier is dependent on the number of attributes and the number of instances. Since we filter the attributes and keep the number of attributes constant at 5000, the learning time is directly proportional to the number of instances. There can be a large number of instances but only a small number of them are useful for learning. So it is useful to filter out the instances that are not useful for learning.

## 6.2 Data Imbalance in IE Datasets

Representing IE as a classification task gives rise to learning tasks with a high degree of imbalance. This gives rise to the behaviour that we see at level 1 where we have high precision but low recall. The fact that we have a high number of negative instances and a small number of positive instances when learning means that the learner is much more likely to predict negative instances than positive instances. For an instance to be predicted as positive, there must be strong evidence that it is a positive instance. Thus the imbalanced models are more likely to make errors that are false-negatives than false-positives.

Data imbalance explains the behaviour that we see at L1 with ELIE. With L1 learning there is a much larger number of negative instances than positive instances. Because of the reasons described in the previous section, the L1 learner is more likely to predict a negative instance that a positive instance. If a positive instance is predicted it is because the evidence that it was positive was strong. Instances that are close to the boundary between classes are more likely to be classified as negatives. Thus when we make a prediction for a positive instance it is likely to be correct. Thus we have high precision at L1 but low recall as many positive instances that are close to the boundary are incorrectly classified as negative.

The two-level approach described in chapter 7 reduces the imbalance in a tar-

geted way and this leads to improved recall while sacrificing some precision.

However we can apply instance filtering techniques to the 1-level IE system (ELIE$_{L1}$). Removing instances can improve execution speed because we have less data to learn from. It can also improve performance by reducing the data imbalance and decreasing the likelihood of producing false negatives.

The datasets that we use for experiments exhibit a large degree of imbalance. The Reuters Corporate Acquisitions dataset has an imbalance of approximately 100:1, the Seminar Announcements dataset 200:1, the job postings 500:1 and the Pascal CFP 900:1. It is interesting to note that ELIE'S performance in relation to other IE algorithms is proportional to the level of imbalance in the dataset, i.e. as the level of imbalance increases ELIE's ability to outperform other systems falls.

## 6.3 Instance Filtering Techniques

### 6.3.1 Random Negative Instance Filtering

The first technique is a simple random negative instance filter. All positive instances are preserved, while a predefined percentage of the negative instances are randomly removed. The advantage of this method is that is easy to implement and efficient to execute. There are a lot of instances in the data that are very uninformative and contribute nothing to the learned model. However there are also informative negative instances and this method can remove these too. The performance of this technique can vary widely over different runs on the same data. If it doesn't remove any very informative negative instances then its performance can be very good. However if it deletes a number of informative negative instances while keeping uninformative ones it can perform very poorly.

Figure 6.2 shows performance (f-measure) of this approach on the Seminar Announcements dataset. The vertical axis gives the f-measure while the horizontal axis gives the percentage of negative instances that were randomly removed. This experiment follows the same methodology as the experiments described in chapter 4. For each field, performance is slightly higher initially but then begins to fall as the filtering rate increases and falls rapidly when the filter rate is high. For *speaker*, *stime* and *etime* there is no appreciable difference in performance

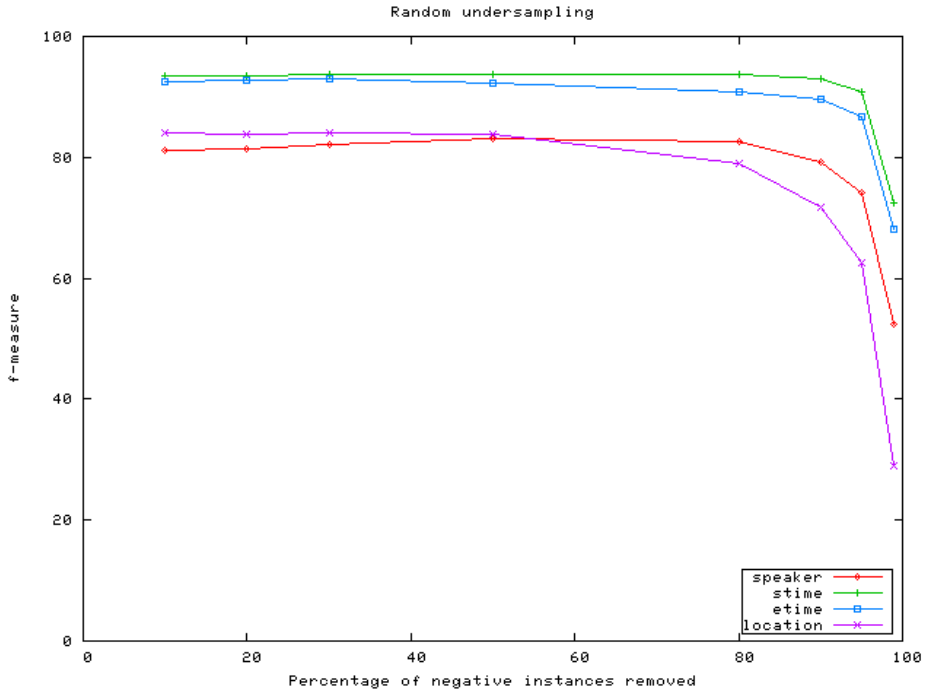


Figure 6.2: Random negative instance filtering: F-measure

with the filtering rate set to 80% than with no filtering. For *location* performance begins to fall once the filtering rate goes above 50%. For *speaker* and *location*, performance begins to fall rapidly once the filtering rate goes above 80% while for *stime* and *etime* the performance falls rapidly once the filtering rate goes above 90%. This indicates that for this dataset, random negative instance filtering is a good way to reduce execution time without harming performance. Learning time is a direct function of the number of instances. We can safely delete up to 50% of the negative instances using this technique. Once the filtering rate goes above 50% performance begins to degrade and above 80% performance degrades significantly.

Figures 6.6 and 6.7 show precision and recall for random undersampling on the Seminar Announcements dataset. When we consider precision and recall separately we see that the drop in performance comes from a fall in precision more than from a fall in recall. For each of the four fields precision drops sharply as the rate of undersampling increases. For *stime*, *etime* and *location* recall only falls

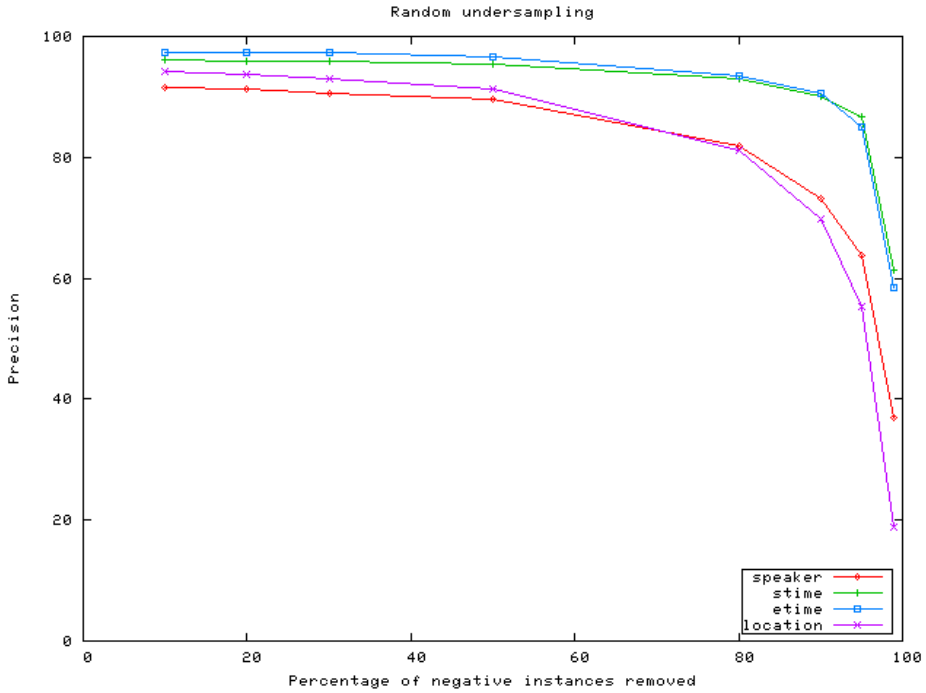


Figure 6.3: Random negative instance filtering: Precision

when the rate of undersampling goes above 95%. For the *speaker* field recall actually rises as the rate of undersampling increases. This confirms that precision is proportional to the level of imbalance in the dataset. By altering the imbalance in the dataset we can alter the precision of the extractor while the recall of the system stays relatively constant.

### 6.3.2 Removing Instances with Uninformative Words

Random filtering can arbitrarily improve accuracy and decrease execution time without hurting performance. But it is not reliable. We need a more reliable method. We wish to remove instances based on how useful they are for learning. We would like to remove uninformative instances and keep informative instances.

One approach to this is to remove instances whose token is uninformative with respect to the target class. Very frequent words are usually uninformative. Removing the most frequently occurring words can reduce the dataset size without

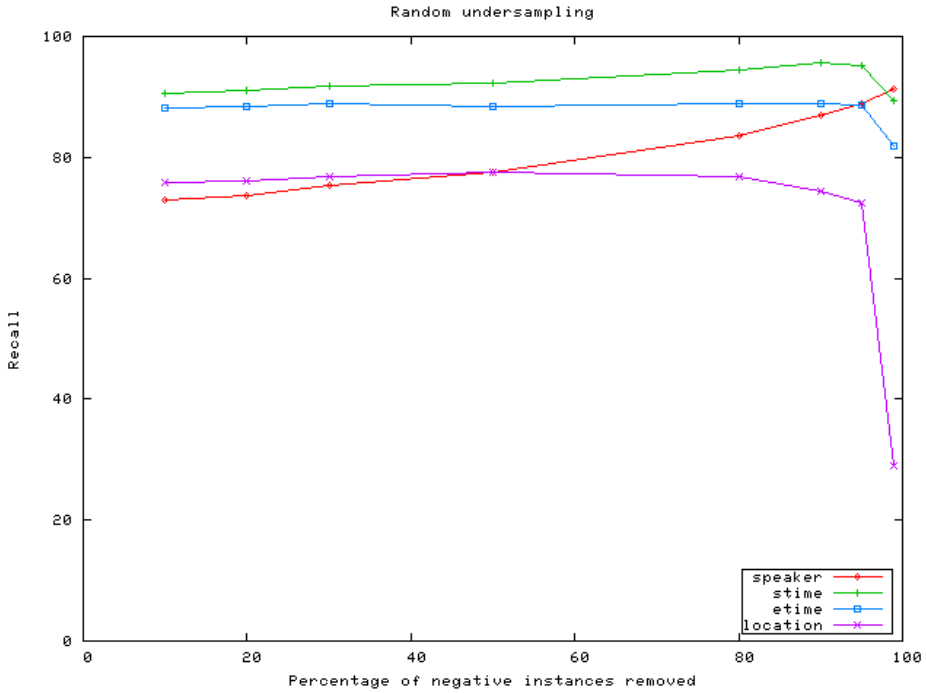


Figure 6.4: Random negative instance filtering: Recall

the risk of removing informative instances. The approach we take is based on that taken by Gliozzo *et al.* [22].

For a corpus *C*, |*C*| is the number of tokens in *C* and $V_c$ is the vocabulary of the corpus *C*. *OCC(w,C)* is the number of occurrences of *w* in *C*.

The probability of a word occurring is

$$p(w) = \frac{OCC(w, C) + 1}{|C| + |V_c|}$$

The set of uninformative words is then given by

$$U_\theta = \{w | p(w) > \theta \; and \; w \in V_C\}$$

The aim is to identify words that are frequently marked as positive examples. Let $OCC^+(w, C)$ and $OCC^-(w, C)$ be the occurrence of the word *w* in the positive and negative examples. Then the probability of the word being in a positive

example is

$$p^+(w) = \frac{OCC^+(w, C) + 1}{|C| + |V_c|}$$

and the probability of the word being in a negative example is

$$p^-(w) = \frac{OCC^-(w, C) + 1}{|C| + |V_c|}$$

Thus the set of uninformative words is given by

$$U_{\alpha,\theta} = \left\{ w \mid \ln \frac{p^-(w)}{p^+(w)} - Z_{1-\alpha} \sqrt{\frac{1}{OCC^+(w, C)} + \frac{1}{OCC^-(w, C)}} \geq \Theta \right\} \quad (6.1)$$

A word is filtered if the odds ratio between $p^+$ and $p^-$ exceeds some predefined threshold. $Z_{1-\alpha}$ is a confidence coefficient measured from statistical tables.

From here our approach diverges from that of Gliozzo *et al.* They perform further steps. They do an exhaustive search of possible values for $\alpha$ and $\theta$. An upper bound $\varepsilon$ is set on the number of positive instances that can be filtered and $\alpha$ and $\theta$ are selected such that they filter the maximum number of negatives while ensuring that the positive filter rate does not exceed $\varepsilon$.

Our approach does not do an exhaustive search for values of $\alpha$ and $\theta$. Instead we remove a fixed, pre-defined percentage of the negative instances. We do not filter any of the positive instances. We fix the value of $\alpha$[2] and calculate $\theta$ for all terms in the vocabulary according to equation 6.2.

$$\theta = \ln \frac{p^-(w)}{p^+(w)} - Z_{1-\alpha} \sqrt{\frac{1}{OCC^+(w, C)} + \frac{1}{OCC^-(w, C)}} \quad (6.2)$$

We then rank all words according to $\theta$ and set a threshold for $\theta$ such that it covers a fixed percentage of the instances and delete all negative instances that have a value for $\theta$ that is below this threshold. E.g. we may set our threshold so that deleting all instances whose token has $\theta$ less than the threshold would result in us deleting approximately 50% of the negative training instances.

[2]We fix $\alpha$ to give 99% confidence

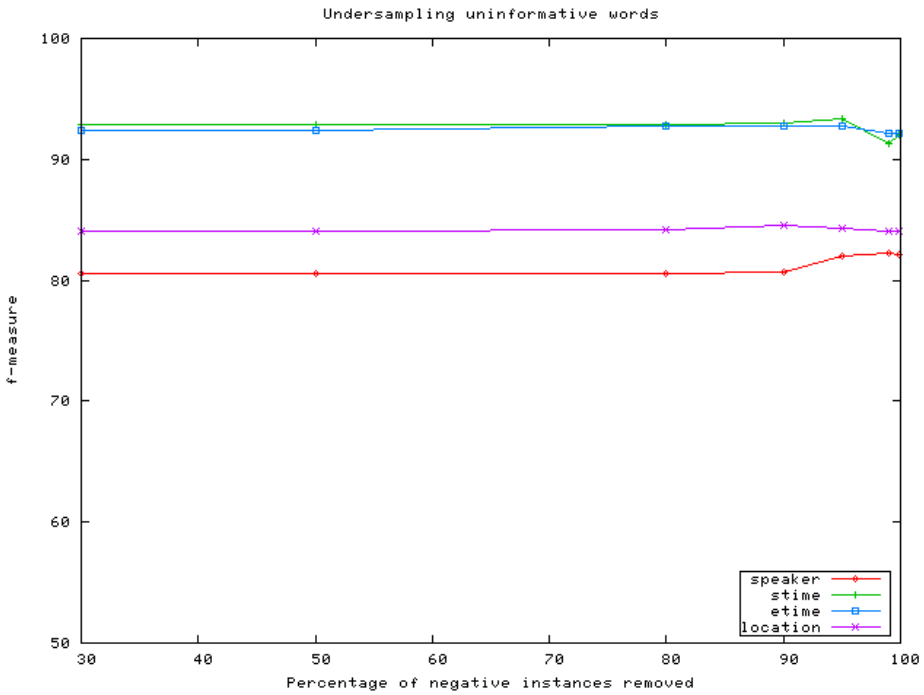


Figure 6.5: Filtering instances with uninformative words: F-measure

We also calculate $\theta$ over all fields that we want to extract, rather than over one field. A token is counted as occurring in a positive instance for all of the fields we wish to extract. Thus the instances that are deleted are the same for all fields. This allows us to keep a single representation of the dataset and we only have to perform the instance filtering once. It also means that we keep instances that are informative for one field, but may be uninformative for other fields. If we were to perform the instance filtering on a per-field basis, we would only keep instances relevant to the particular field in question but we would need to perform the filtering several times. Doing it once is an approximation for simplicity and efficiency. Thus there is an upper bound on the number of negative instances that are filtered. If we are extracting more than one field, there will always be negative instances in the dataset as instances that are positives for one field will be negatives for other fields.

We performed some experiments on the Seminar Announcements dataset to investigate this instance filtering approach. Figure 6.5 shows performance on this

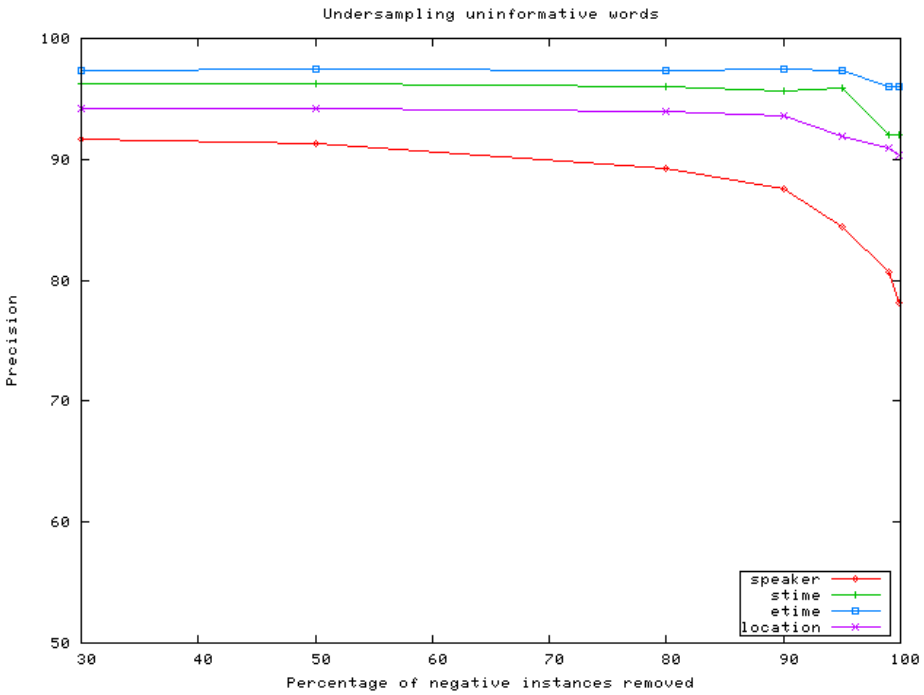


Figure 6.6: Filtering instances with uninformative words: Precision

dataset using this filtering technique. The vertical axis shows f-measure while the horizontal axis shows the percentage of negative instances filtered. Performance is relatively constant as the filtering rate increases. Even with high filtering rate the performance remains high. This indicates that this technique can filter a large number of negative instances without harming performance. This technique removes instances that are not likely to occur in the positive examples. Thus they are not likely to be close to the boundary so removing them doesn't adversely affect the SVM performance.

F-measure gives the weighted average of precision and recall. Since f-measure is relatively stable with increased filtering we must investigate if precision and recall are changing as the filtering rate increases.

Figure 6.6 shows precision for the same experiment. It shows that precision falls for all four fields as the filtering rate increases. As the filtering rate goes above 90% the drop in precision increases significantly. The *location* and *speaker* fields have a higher fall in precision than *stime* and *etime*. *Speaker* and *location*

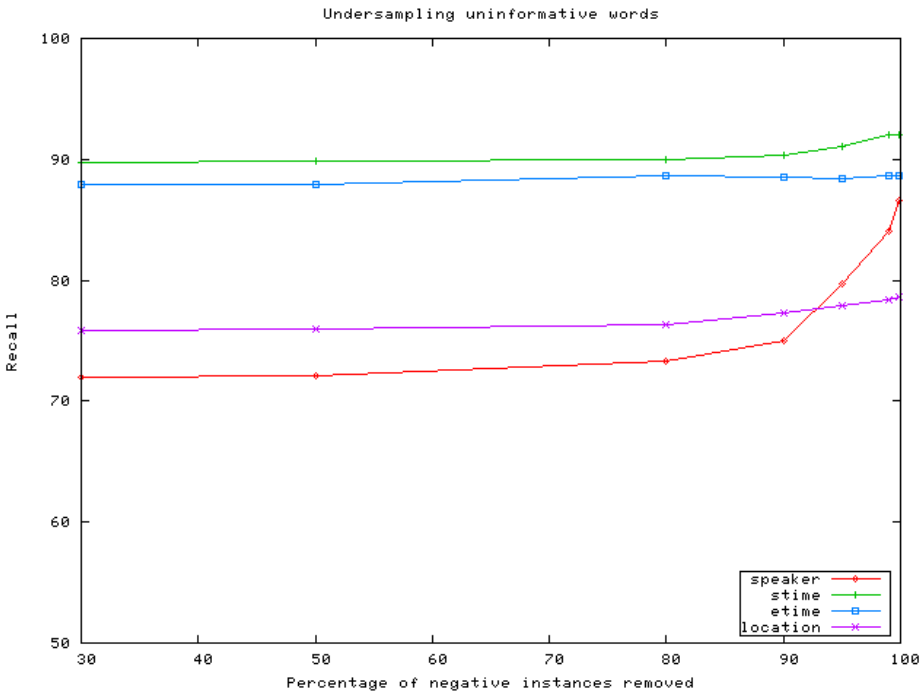


Figure 6.7: Filtering instances with uninformative words: Recall

are the more difficult fields to learn so they probably require more instances to learn well.

Figure 6.7 shows the recall on the Seminar Announcements. The *speaker* field shows a large increase in recall as the filtering rate increases. The *location* and *stime* fields show small increases in recall as the filtering rate increases while etime is fairly constant.

In general precision falls as more instances are filtered and recall increases. This increase in recall cancels out the fall in precision and the f-measure stays relatively constant. Undersampling pushes the boundary closer to the negative instances and away from the positive instances. This gives an increase in recall as we are more likely to identify positive instances. It may be the case that undersampling also changes the angle of the hyperplane, as described in [3], resulting in lower precision.

## 6.4 Discussion

There are two reasons we wish to remove negative instances. The first is to improve execution time. Execution time is a direct function of the number of instances. Many of the negative instances do not contribute to our ability to learn our target concept. We would like to delete negative instances that are not informative to reduce the time it takes to learn. For SVMs these uninformative instances are instances that are not close to the boundary between classes. We examined two methods for instance filtering. Both methods can remove some of these instances without harming accuracy. Random undersampling is useful as long as the filter rate is not too high. The higher the filter rate, the higher the likelihood of removing informative instances and reducing the accuracy of the learning algorithm. Filtering uninformative instances is effective even at high rates of filtering. It causes precision to fall slightly, but recall rises and f-measure stays constant even with very high levels of filtering.

The second reason to remove some of the negative instances is to address the class imbalance problem. Removing negative instances may move the hyperplane back towards the negative examples leading to greater recall.

Neither of these instance filtering techniques really address the class imbalance problem. If they did we would see performance rising as more instances are removed. We do see some small increase in performance when filtering uninformative instances at very high levels of filtering. To better address the class imbalance problem would require us to remove all the uninformative instances, but to also remove some of the negative instances that are close to the hyperplane without hurting performance.

## 6.5 Summary

In this chapter we described two approaches to instance filtering. There are two reasons to perform instance filtering. The first it to reduce execution time without harming accuracy by removing instances that are not informative for learning. The second is to address the problem of class imbalance in IE and reduce the class imbalance to improve accuracy.

- We described two instance filtering methods and assessed their impact on the performance of the system.
- An approach that removes random negative instances can reduce execution time without harming accuracy up to a certain threshold which in our experiments was 50%. With this random approach there is a chance of filtering informative instances and this increases as the rate of filtering increases.
- We also showed that an approach that removes instances with tokens that are unlikely to occur in the positive examples can filter a large proportion of the negative instances without harming accuracy.
- The precision of the extraction system is proportional to the level of imbalance in the data. High imbalance gives high precision. With random undersampling precision drops much more than recall as the level of undersampling increases.
- Neither approach significantly improved accuracy but both allow us to filter instances for improved execution time. The approach that filters uninformative tokens is much more reliable at higher filtering levels.

# Chapter 7

# A Two-level classification approach to Information Extraction

In this chapter we extend the one-level classification approach to IE that we described in chapter 4. We showed that the one-level approach is competitive with other current state of the art IE systems. We extend this approach by adding a second level of more focused classifiers that use the predictions of the one-level approach as a guide for their own predictions. This second layer of focused classifiers allows us to improve the recall while still maintaining good precision. We will call this approach ELIE$_{L2}$ (abbreviated to L2).

## 7.1 A Two-level Approach to Learning

The L1 learner builds its model based on a very large number of negative instances and a small number of positive instances. Therefore the prior probability that an arbitrary instance is a boundary is very small. This gives a model that has very high precision. Because the prior probability of predicting a tag is so low, and because the data is highly imbalanced, when we actual do predict a tag, it is very likely that the prediction is correct. The L1 model is therefore much more likely to produce false negatives than false positives (high precision).

The intuition behind the two-level approach is as follows. At L1, the start and end classifiers have high precision. To make a prediction, both the start classifier

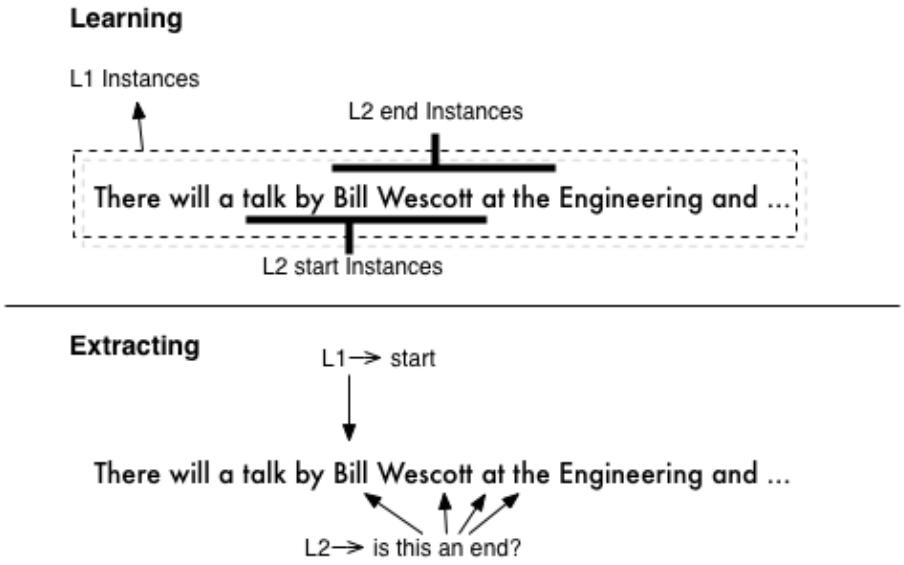


Figure 7.1: L1 and L2: An example

and the end classifier have to predict the start and end respectively. In many cases where we fail to extract a fragment, one of these classifiers made a prediction, but not the other. Level 2 assumes that these predictions are correct and is designed to identify the starts and ends that we failed to identify at level 1.

The L2 models are learned from training data in which the prior probability that a given instance is a boundary is much higher than for the L1 learner. This "focused" training data is constructed as follows. When building the L2 start model, we take only the instances that occur a fixed distance before an end tag. Similarly, for the L2 end model, we use only instances that occur a fixed distance after a start tag.

For example, an L2 window of size 10 means that the L2 start model is built using only all those groups of 10 instances that occur before an end-tag in the training data, while the L2 end model is built using only those instances that occur in the 10 instances after a start tag in the training data. Note that these L2 instances are encoded in the same way as for L1; the difference is simply that the L2 learner is only allowed to look at a small subset of the available training data.

Fig. 7.1 shows an example of the instances used by L1 and L2 with a looka-

head/lookback of 3. In this example the token 'Bill' is the start of a field and the token 'Wescott' is the end of a field. When building the L1 classifiers we use all the available instances. When building the L2 start model we use the end token and the 3 tokens preceding it. When building the end model we use the start token and the three tokens following it. Note that these L2 instances are encoded in the same way as for L1; the difference is simply that the L2 learner is only allowed to look at a small subset of the available training data. When extracting, the L2 end classifier is only applied to the three tokens following the token which L1 tagged as a start and the token itself. Similarly the L2 start classifier is only applied to instances tagged as an end by L1 and the three preceding tokens.

This technique for selecting training data means that the L2 models are likely to have much higher recall but lower precision. If we were to blindly apply the L2 model to the entire document, it would generate a lot of false positives. Therefore, as shown in Figures 7.1 and 7.3, the reason we can use the L2 model to improve performance is that we only apply it to regions of documents where the L1 model has made a prediction. Specifically, during extraction, the L2 classifiers use the predictions of the L1 models to identify parts of the document that are predicted to contain fields.

Figure 7.2 summarizes the learning process for this two-level learner (which we call ELIE$_{L2}$). As before (with ELIE$_{L1}$) the training documents are converted into a set of training instances that are used to build the L1 models and a set of start/end pairs used to build the histogram for the tag-matcher. There is then a combination and reduction step to get the training instances used by L2 from those used by L1. The training instances from L1 are reduced in that we only use instances that occur a fixed distance after a start or before an end. They are combined in that we use the L1 start instances to pick the L2 end instances and we use the L1 end instances to pick the L2 start instances. Once we have identified the L2 start and end training instances we use them to build the L2 start and L2 end models.

Figure 7.3 summarizes the extraction process for the two-level approach. Given a set of documents that we want to extract from, we convert these documents into a set of instances. We apply our L1 models for start and end to these instances and generate a set of predictions for starts and ends. The L1 predictions are then used

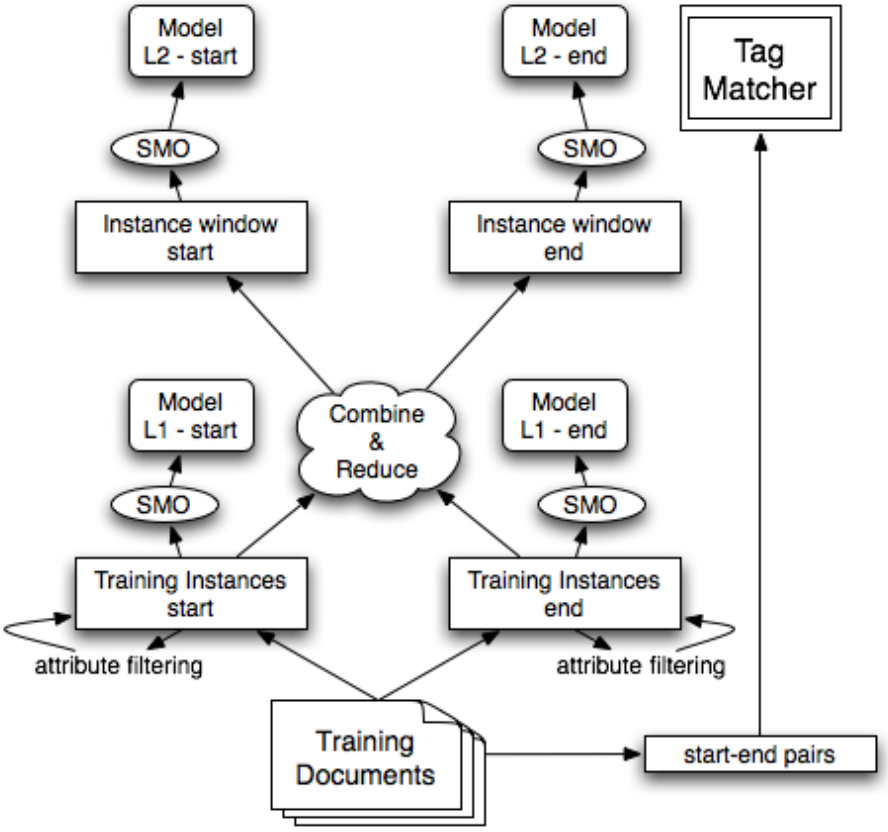


Figure 7.2: L2 Learning: Overview

to guide which instances we apply the two-level classifiers to. We use the predictions of the L1-end model to decide which instances to apply the L2-start model to, and we use the predictions of the L1-start model to decide which instances to apply the L2-end model to. Applying the L2 models to the selected instances gives us a set of predictions which we pass to the tag-matcher to get our extracted fields.

The L2 approach is based on the assumption that L1 has high precision but that recall could be improved. In order to extract a fragment, we must identify both its start and its end. However in many cases we identify the start of a fragment but not the end or vice-versa. In these cases we fail to extract the fragment.

The start and end classifiers are likely to have high precision because of the

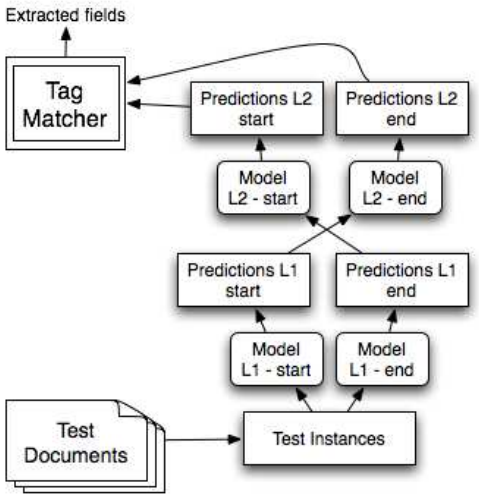


Figure 7.3: L2 Extracting: Overview

imbalance in the data. We assume that the L1 predictions are high precision and that if we predict a start or an end it is correct. Thus if we identify a start but failed to identify the corresponding end, then there is likely to be an unidentified end in the tokens following the start.

The intuition behind the two-level approach is that we use the unmatched L1 predictions (i.e. when we identify either the start or the end but not the other) as a guide to areas of text that we should look more closely at. We use more focused classifiers that are more likely to make a prediction on areas of text where it is highly likely that an unidentified fragment exists. These classifiers are more likely to make predictions due to a much lower data imbalance so they are only applied to instances where we have high probability of a fragment existing.

As the level of imbalance falls, the recall of the model rises while precision falls. We use a L2 lookahead/lookback of 10 for our experiments so the imbalance of the L2 data is approximately 10:1. This is much lower than the imbalance in the L1 classifiers which can anything from 100:1 to 1000:1 depending on the dataset.

This enables us to improve recall without hurting precision by identifying the missing complementary tags for orphan predictions. If we have 100% precision at L1 then we can improve recall without any corresponding drop in precision. In practice, the drop in precision is proportional to the number of incorrect predic-

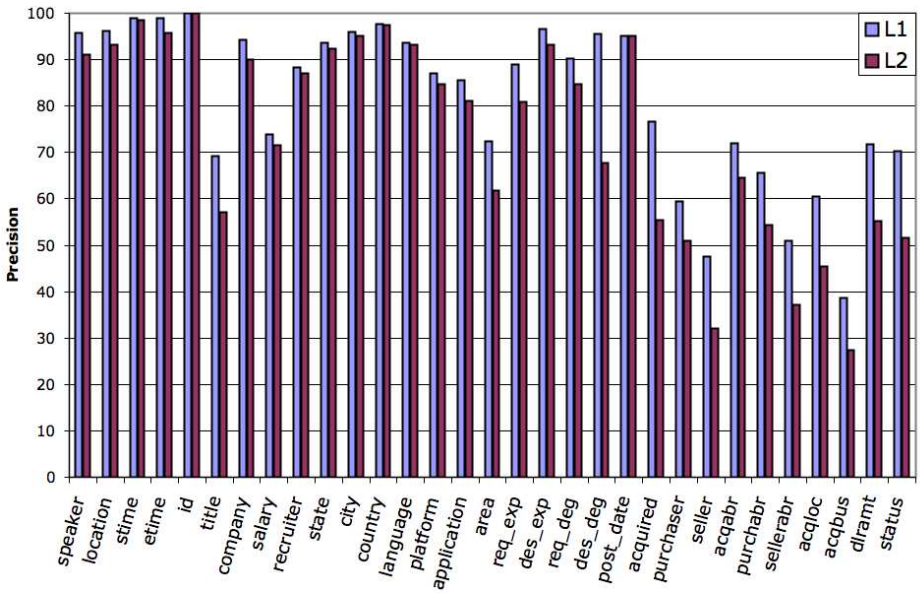


Figure 7.4: Comparing L1 to L2: Precision

tions at L1.

## 7.2 Evaluation

We evaluated this two-level approach on the three benchmark IE datasets using the same methodology as previously.

### 7.2.1 Comparing L2 to L1

Figure 7.4 shows the Precision of this two-level approach compared to the precision of the one-level approach on the three benchmark datasets. On almost all of the fields, L2 has lower precision than L1.

Figure 7.5 compares the recall of L1 and L2. In contrast to precision, recall increases between L1 and L2 for all fields. The L2 approach results in higher recall.

Figure 7.6 shows the f-measure of the L1 and L2 approaches. We have shown already that precision falls and recall increases when we use L2. We are interested

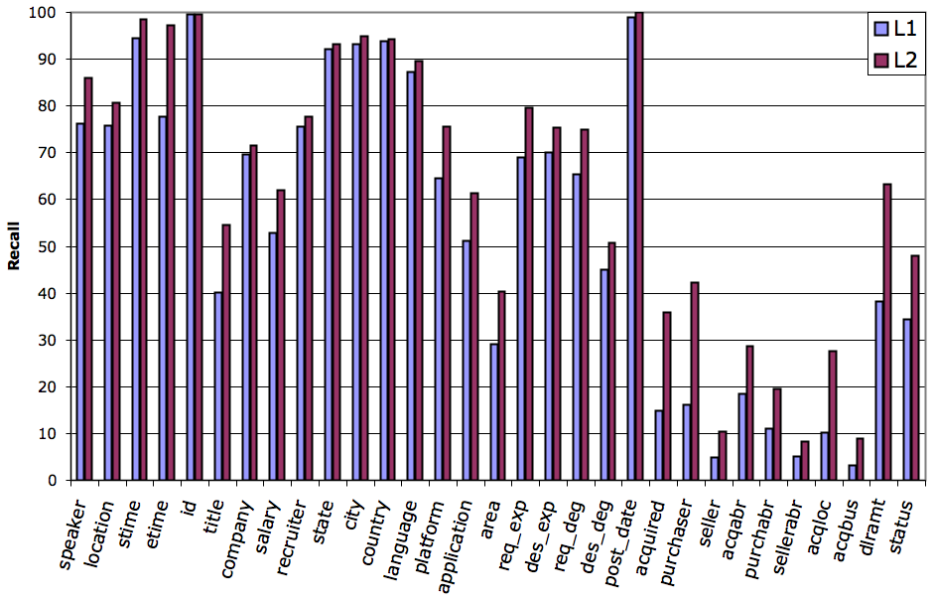


Figure 7.5: Comparing L1 to L2: Recall

in whether the increase in recall in enough to offset the fall in precision when we consider f-measure. F-measure for L2 is higher than for L1 for almost all fields. In fact only one of the 31 fields has higher f-measure at L1. For many fields the increase in f-measure at L2 is substantial.

We would like to determine whether the increase in f-measure at L2 is statistically significant. We cannot do statistical significance testing when we are comparing against the published results of other systems. However because we have controlled the L1 and L2 experiments and all variables in the experiment are equal (so the data is paired), we can use a paired t-test to test for statistical significance.

Figure 7.7 shows the results of the paired t-test for statistical significance for the hypothesis that the f1 at L2 is higher than the f1 at L1 [1]. For all 31 fields, L2 is never statistically significantly worse than L1. On 21 of the 31 fields the L2 approach is statistically significantly better than the L1 approach.

We conclude that the two-level approach gives improved performance over a one-level approach. Adding a second level of biased classifiers often gives sub-

[1]We used a one-tailed test with $\alpha$=0.01.

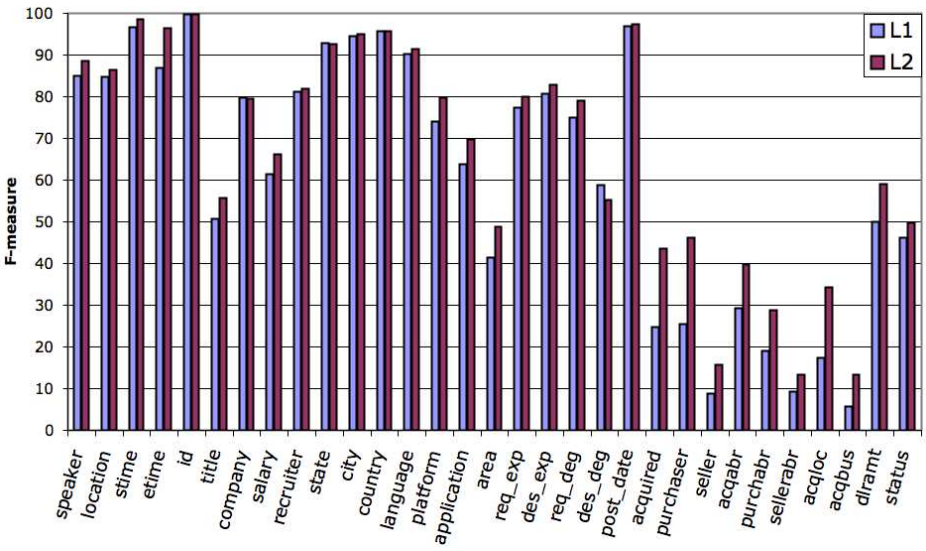


Figure 7.6: Comparing L1 to L2: F-measure

| field | Significant | field | Significant |
|---|---|---|---|
| speaker | yes | req_exp | yes |
| location | no | des_exp | no |
| stime | no | req_degree | yes |
| etime | yes | des_degree | no |
| id | no | post_date | no |
| title | yes | acquired | yes |
| company | no | purchaser | yes |
| salary | yes | seller | yes |
| recruiter | no | acqabr | yes |
| state | no | purchabr | yes |
| city | yes | sellerabr | yes |
| country | no | acqloc | yes |
| language | yes | acqbus | yes |
| platform | yes | dlramt | yes |
| application | yes | status | yes |
| area | yes | | |

Figure 7.7: F1 at L2 is statistically significantly better than F1 at L1

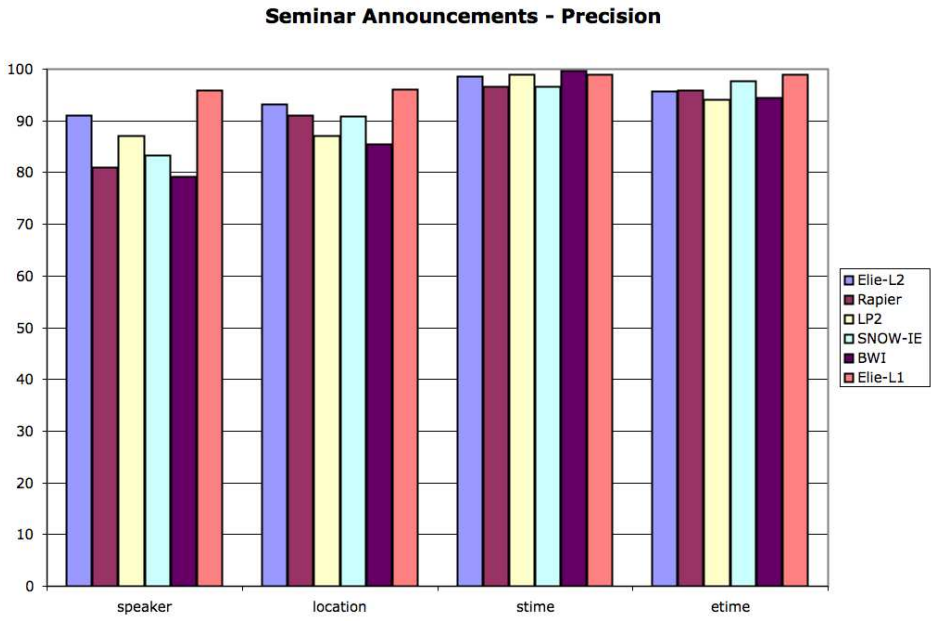


Figure 7.8: L2 Precision for the Seminar Announcements dataset

stantial improvement in performance and never degrades performance. L2 gives lower precision but compensates for this by giving an even larger increase in recall.

We saw in chapter 6 that undersampling uninformative tokens caused precision to fall and recall to rise. But the fall in precision was offset by the rise in recall so f-measure remained relatively constant. With this two-level approach the increase in recall is greater than the fall in precision. Thus f-measure is generally higher and never lower than the one-level approach.

### 7.2.2 Comparing L2 to other IE systems

We compared the two-level approach to the other IE systems.

Figure 7.8 shows the precision on the Seminar Announcements dataset of ELIE$_{L2}$ compared with other IE systems. Precision for ELIE$_{L2}$ is lower than for ELIE$_{L1}$ on all 4 fields. Even though precision for L2 is lower than for L1, it is still competitive with each of the other algorithms. It has second-best precision for two of the 4 fields and is close to best for the other two.

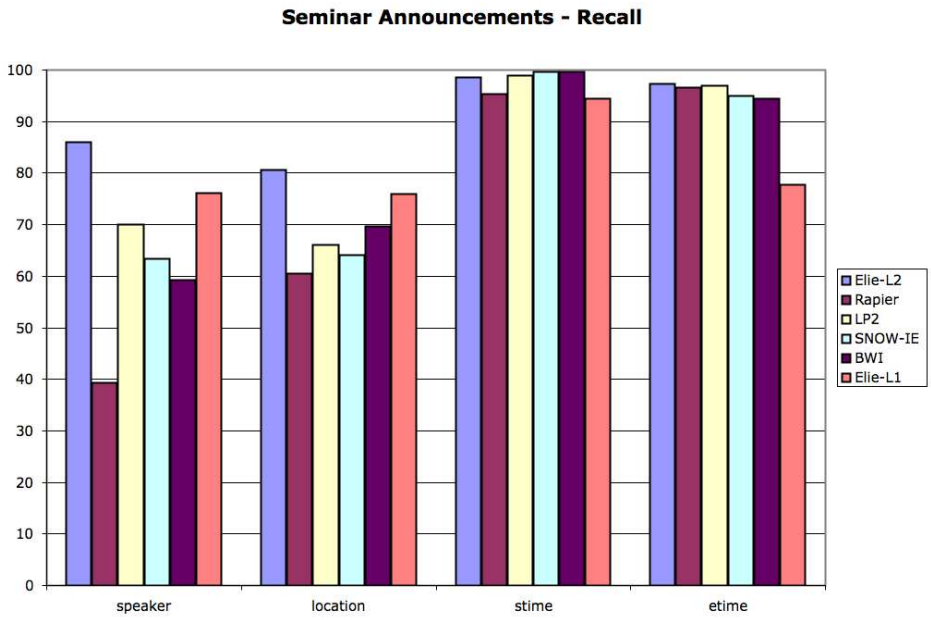


Figure 7.9: L2 Recall for the Seminar Announcements dataset

Figure 7.9 shows recall for ELIE$_{L2}$ on the Seminar Announcements dataset. ELIE$_{L2}$ has the highest recall of any of the systems on 3 of the 4 fields. Recall for *speaker* and *location* is much higher than for any of the other systems.

Figure 7.10 shows f-measure for the Seminar Announcements dataset. ELIE$_{L2}$ has significantly higher f-measure than any of the competitor systems on the *speaker* and *location* fields (these are the more difficult fields). It is also among the top performing systems on the other two fields.

Figure 7.11 shows precision for the Job Postings dataset compared to other IE systems. In general ELIE$_{L2}$ is competitive with the other IE systems although it is outperformed by ELIE$_{L1}$. On the *salary*, *application*, *area* and *desired degree* fields ELIE$_{L2}$ has the worst precision of any of the systems. On some of the other fields it is among the best performing while in general it is in the middle.

Figure 7.12 shows recall for the Job Postings dataset. ELIE$_{L2}$ is the top performing system on several fields and is among the top performing systems on most fields.

Figure 7.13 shows f-measure for the Job Postings dataset. ELIE$_{L2}$ is the top performing system on 6 of the 17 fields. It is among the top performing systems

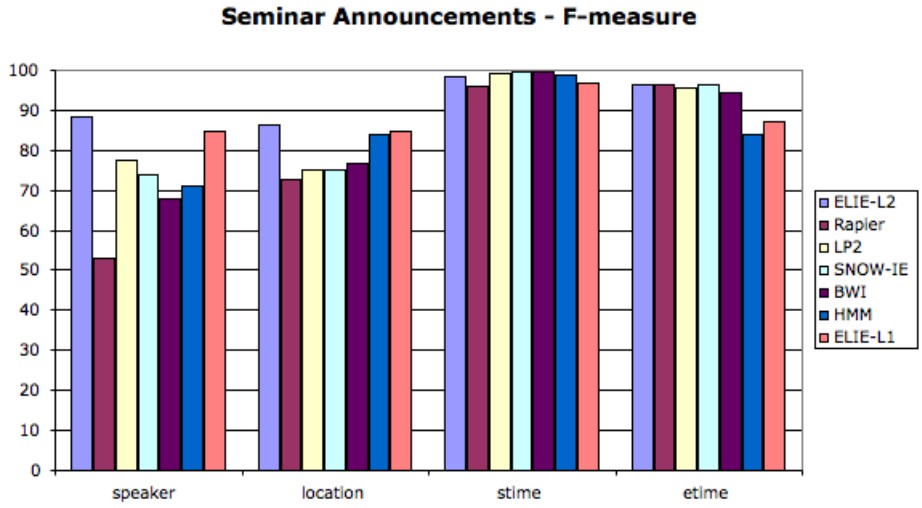


Figure 7.10: L2 F-measure for the Seminar Announcements dataset

on several other fields. It is the worst performing system on one of the fields (desired degree). $(LP)^2$ is the best performing system on 4 fields and worst on 1 field. The results for ELIE$_{L2}$ on multi-valued fields (e.g. *platform*, *application*, *area*) are probably understated due to the conservative way that we evaluate our system. The other systems probably use a form of OSO evaluation. For fields that occur many times within documents the difference between the OSO and ASO evaluations can be large.

Figure 7.14 shows precision on the Reuters Corporate Acquisitions dataset. ELIE$_{L2}$'s precision is competitive with Rapier on this dataset. This is a difficult dataset for extraction and performance is lower than on other datasets. ELIE$_{L2}$'s performance indicates that when performance is poor, ELIE$_{L2}$ has the potential to make substantial improvements.

Figure 7.15 shows recall and figure 7.15 shows f-measure for ELIE$_{L2}$ for the Corporate Acquisitions dataset. ELIE$_{L2}$ is the best performing system on most of the fields. On most fields its performance is substantially better than Rapier and HMM. HMM matches it one field and outperforms it on one field.

Figure 7.17, Figure 7.18 and Figure 7.19 summarize the performance of ELIE$_{L2}$ against that of other IE systems. The horizontal axis shows the performance of ELIE$_{L2}$ while the vertical axis shows the performance of the competitor system.

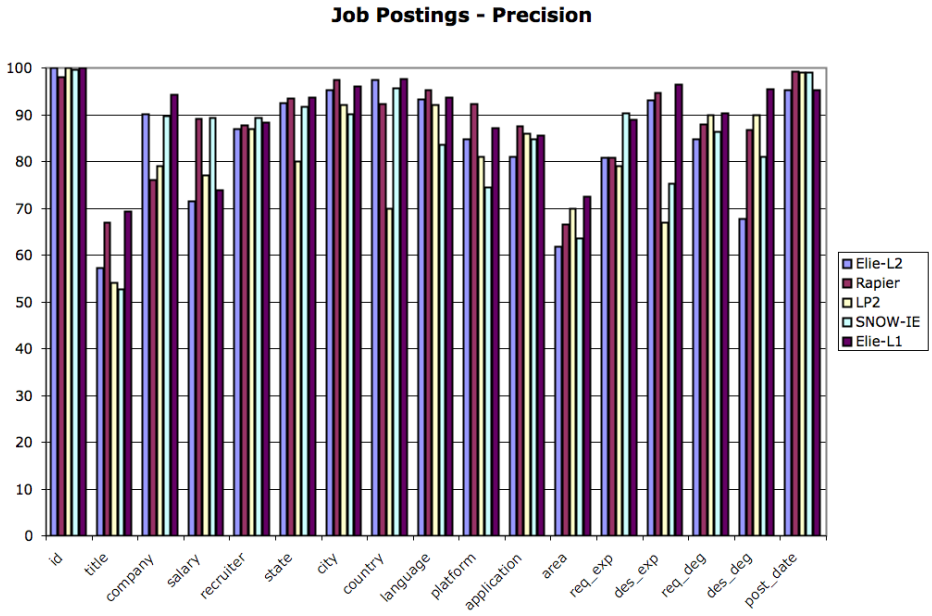


Figure 7.11: L2 Precision for the Job Postings dataset

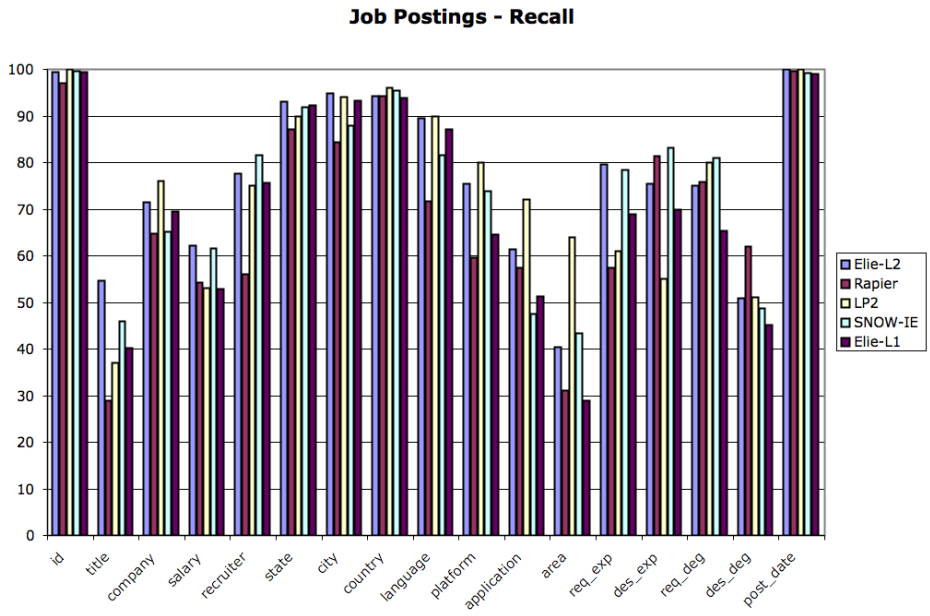


Figure 7.12: L2 Recall for the Job Postings dataset

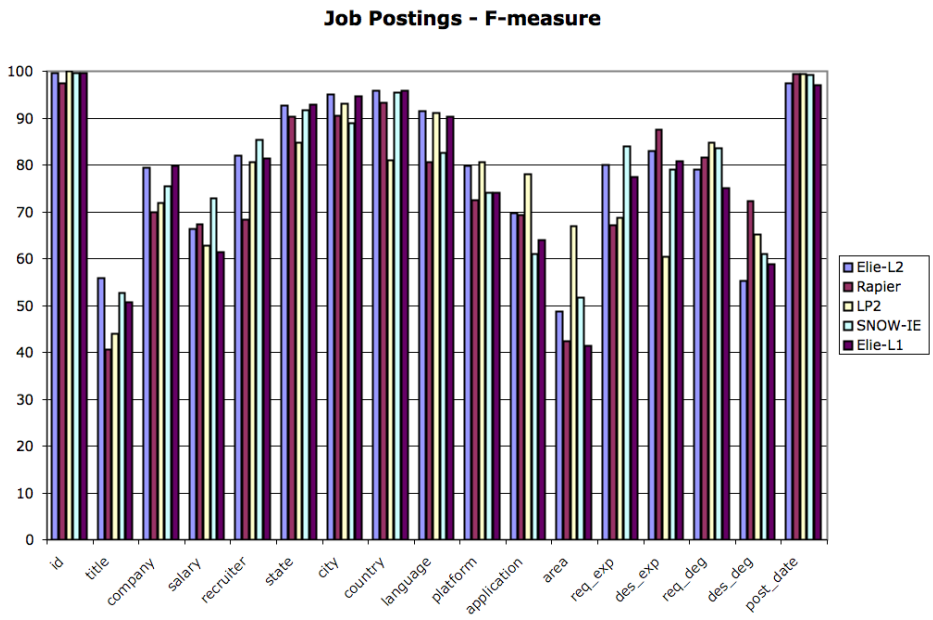


Figure 7.13: L2 F-measure for the Job Postings dataset

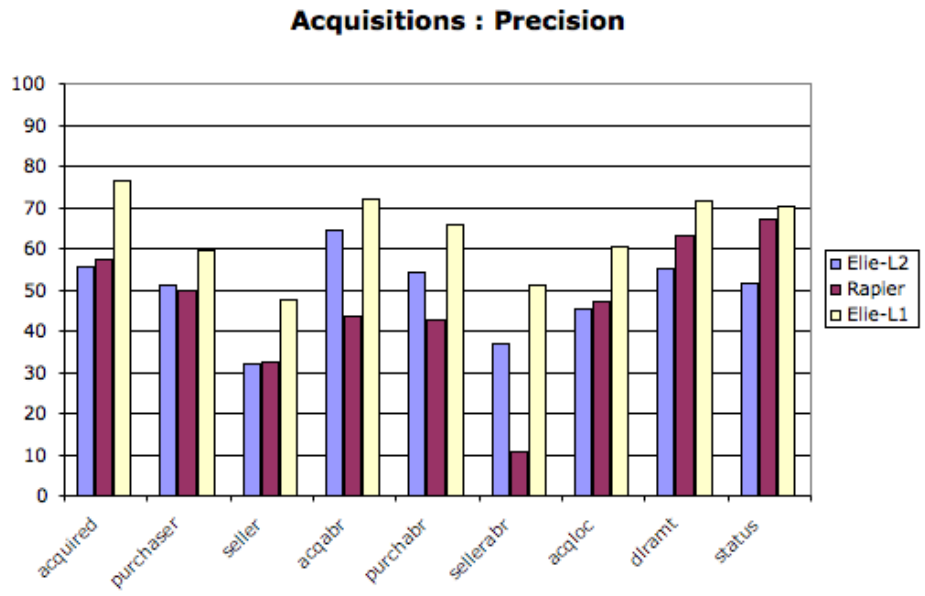


Figure 7.14: L2 Precision for the Reuters Corporate Acquisitions dataset

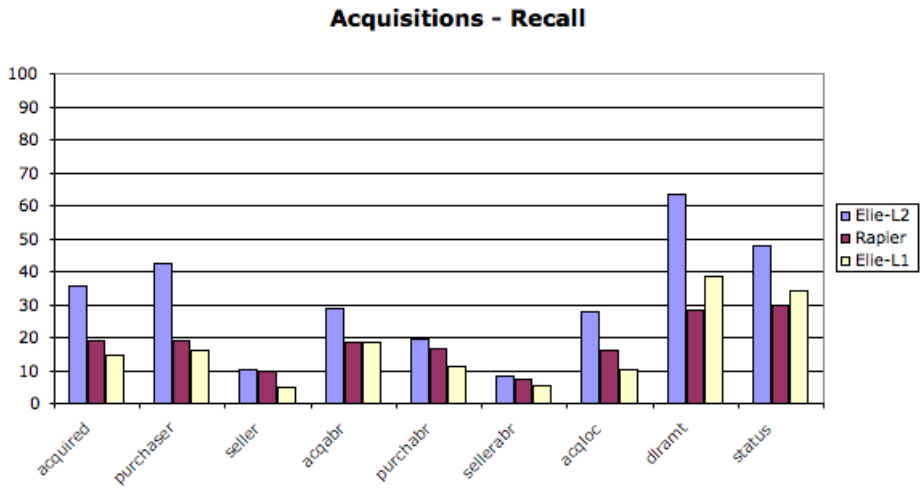


Figure 7.15: L2 Recall for the Reuters Corporate Acquisitions dataset

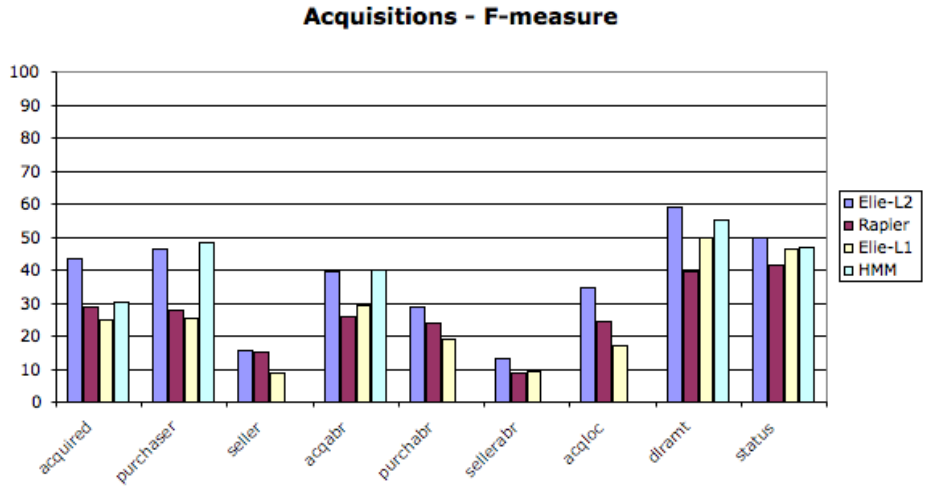


Figure 7.16: L2 F-measure for the Reuters Corporate Acquisitions dataset

Each point represents the performance of ELIE$_{L2}$ vs a competitor on a single field. Points that occur above the diagonal line indicate that the competitor system is doing better while points occurring below the diagonal line indicate that ELIE$_{L2}$ is doing better.

On the precision graph, the points are clustered around the diagonal. For recall and f-measure, the majority of points are below the diagonal or clustered close to

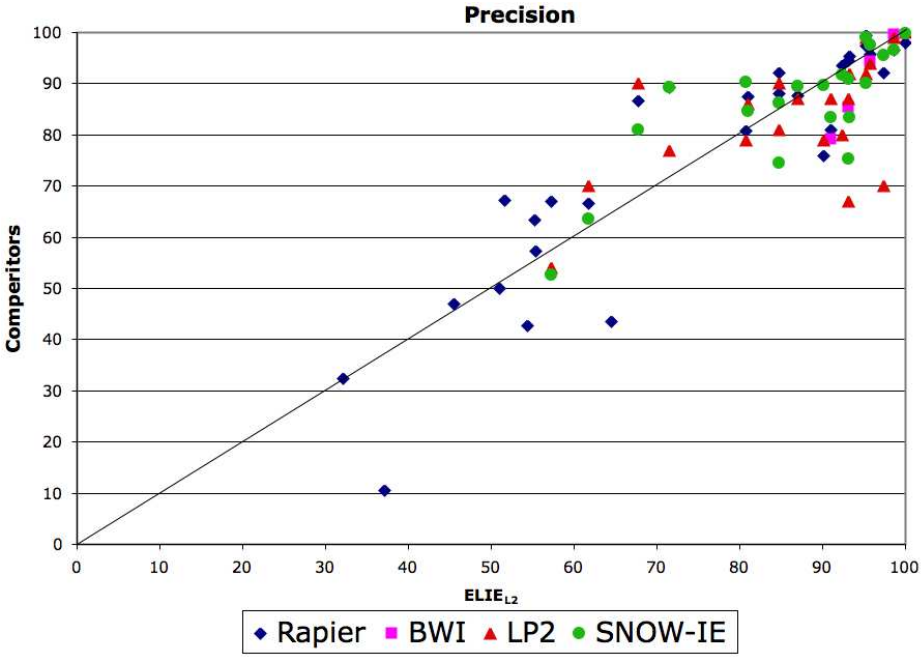


Figure 7.17: L2 Precision summary

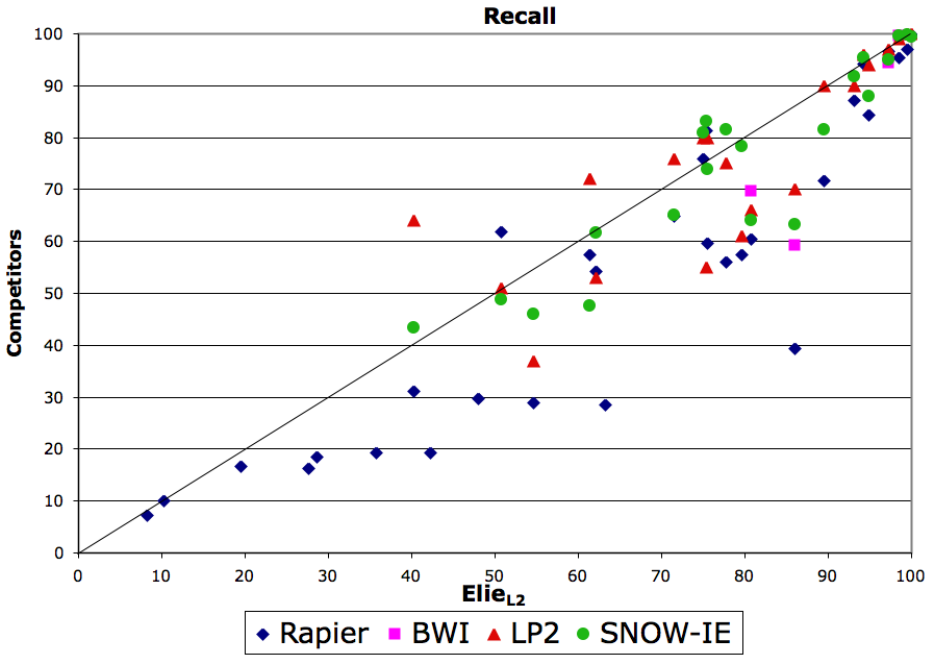


Figure 7.18: L2 Recall summary

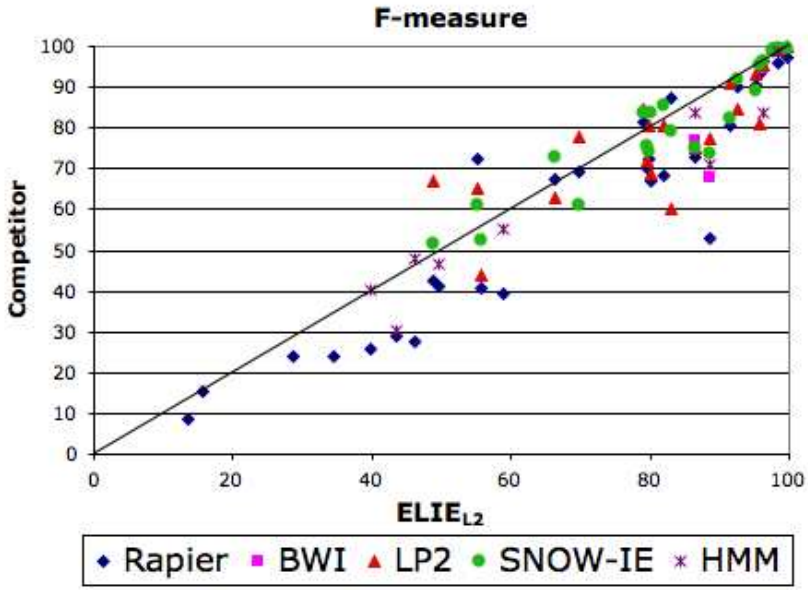


Figure 7.19: L2 F-measure summary

it. We conclude that ELIE$_{L2}$ generally outperforms the other IE systems on the three benchmark IE tasks.

## 7.3 Discussion

The instance filtering techniques described in chapter 6 can significantly improve execution time without affecting performance but they do not significantly improve the accuracy of the classifiers. Random undersampling can improve or hurt performance depending on whether it deletes instances that are informative or not. The second instance filtering technique deletes instances that contain tokens that are uninformative with respect to the positive class. The instances are likely to be instances that are not close to the boundary so that deleting them should not affect accuracy.

Table 7.20 shows details of the errors made by ELIE. For all fields in the three benchmark datasets we show the ratio of false positives to false negatives (FP:FN). It also shows the percentage of false positives that were partially correct and the

| | | L1 | | | L2 | | |
|---|---|---|---|---|---|---|---|
| Dataset | Field | FP:FN | %FP$_{ptl}$ | %FN$_{ptl}$ | FP:FN | %FP$_{ptl}$ | %FN$_{ptl}$ |
| SA | speaker | 0.17 | 22 | 62 | 1.05 | 17 | 8 |
| SA | location | 0.19 | 76 | 67 | 0.51 | 75 | 20 |
| SA | stime | 0.2 | 27 | 86 | 4.72 | 9 | 36 |
| SA | etime | 0.05 | 64 | 92 | 0.93 | 36 | 18 |
| Jobs | id | 0 | 0 | 100 | 0 | 0 | 100 |
| Jobs | title | 0.29 | 71 | 58 | 0.9 | 56 | 23 |
| Jobs | company | 0.14 | 9 | 10 | 0.27 | 14 | 2 |
| Jobs | salary | 0.4 | 76 | 68 | 0.66 | 68 | 43 |
| Jobs | recruiter | 0.41 | 22 | 22 | 0.52 | 21 | 11 |
| Jobs | state | 0.79 | 9 | 24 | 1.11 | 9 | 6 |
| Jobs | city | 0.56 | 1 | 28 | 0.95 | 1 | 1 |
| Jobs | country | 0.36 | 0 | 6 | 0.44 | 0 | 3 |
| Jobs | language | 0.25 | 41 | 45 | 0.52 | 30 | 10 |
| Jobs | platform | 0.27 | 43 | 43 | 0.54 | 37 | 10 |
| Jobs | application | 0.18 | 23 | 27 | 0.38 | 14 | 3 |
| Jobs | area | 0.15 | 34 | 25 | 0.41 | 25 | 6 |
| Jobs | req_exp | 0.28 | 8 | 41 | 0.92 | 6 | 9 |
| Jobs | des_exp | 0.09 | 100 | 10 | 0.23 | 54 | 12 |
| Jobs | req_degree | 0.21 | 0 | 34 | 0.53 | 2 | 1 |
| Jobs | des_degree | 0.04 | 0 | 10 | 0.51 | 5 | 0 |
| Jobs | post_date | 4.8 | 0 | 100 | 0 | 0 | 0 |
| Reuters | acquired | 0.05 | 32 | 32 | 0.45 | 18 | 3 |
| Reuters | purchaser | 0.13 | 10 | 35 | 0.7 | 8 | 3 |
| Reuters | seller | 0.06 | 1 | 6 | 0.24 | 2 | 0 |
| Reuters | acqabr | 0.09 | 9 | 14 | 0.22 | 8 | 1 |
| Reuters | purchabr | 0.07 | 5 | 11 | 0.2 | 8 | 1 |
| Reuters | sellerabr | 0.05 | 1 | 4 | 0.15 | 2 | 0 |
| Reuters | acqloc | 0.07 | 16 | 27 | 0.46 | 16 | 3 |
| Reuters | dlramt | 0.24 | 27 | 53 | 1.39 | 15 | 14 |
| Reuters | status | 0.22 | 23 | 35 | 0.87 | 21 | 8 |

Figure 7.20: ELIE error analysis

percentage of false negatives that were partially predicted.

For a false positive to be partially correct means ELIE extracted a fragment, but that it was correct at only one end (either the start or end was not predicted

exactly). These kinds of predictions are still useful in a practical setting and a less conservative method of evaluation might give some credit for these kinds of errors. On several fields, a large proportions of the errors are of this form.

For a false negative to be partially predicted means that for a fragment that we failed to extract, we predicted either the start or the end correctly, but may not have predicted the other. These are the kinds of errors that facilitate the improvement shown by L2 over L1. In general L2 gives a large reduction in these partial errors.

An investigation of the errors that ELIE produces reveals that at L1 most errors are false negatives. Those that are false positives are mostly of two kinds. The first are as a result of using exact matching for evaluation, where we have tagged one end of the field correctly but not the other. The second occur as a result of labelling errors on the data where we extract something that should have been labelled but was not.

The ratio FP:FN shows that at L1, most of the errors are false negatives, while at L2 we generally see an increase in false positives and a reduction in false negatives. This corresponds with ELIE's observed behaviour of high precision at L1 and high recall at L2.

ELIE$_{L1}$ outperformed the systems that it was compared against on most fields in terms of recall or f-measure. If high precision is required then ELIE$_{L1}$ can be used. We evaluated our system conservatively so its performance may be understated in relation to competitors.

The L2 learner consistently improves recall while keeping precision high. On more difficult fields the improvements are generally larger. The L2 classifier always improves recall and usually keeps precision high enough to improve F1.

It is likely that the accuracy of ELIE overall has several sources. Since the L1 classifier alone often gives better performance than other IE algorithms, we conclude that the use of Support Vector Machines as the learning algorithm and the features that our systems uses gives rise to substantial improvement compared to the specialized learning algorithms and narrower feature-sets used by most IE algorithms. Secondly the two-level classification that we have described can give significant increases in performance. It increases recall while maintaining good precision. In many cases, L2 improves ELIE's L1 performance substantially.

## 7.4 Summary

In this chapter we described a two-level classification approach to IE. This approach added a second phase to the one-level classification approach that was previously described. The second phase is designed to increase recall. We find that often false negatives are "almost" extracted (the start but not the end is extracted or the end but not the start). In the second phase ELIE is trained to detect the end of a fragment given its beginning or the beginning of a fragment given its end.

We evaluated this two-level approach on the three standard IE benchmark datasets. In comparison to L1 it improves recall at the expense of precision. However the drop in precision is smaller than the rise in recall so L2 improves f-measure on nearly all fields. The two level approach can improve recall while maintaining high precision. We showed that the f-measure achieved by the two-level approach was consistently better and never worse that that achieved by the one-level approach. On 21 of the 31 fields that we evaluated it on the two-level approach was statistically significantly better than the one-level approach.

ELIE$_{L2}$ is among the top performing systems of the IE systems that we compared it to. It is the best performing algorithm on the Seminar Announcements and the Reuters Corporate Acquisitions datasets. It is consistently one of the best on the jobs dataset.

# Chapter 8

# The Pascal Challenge

## 8.1 Overview

The Pascal Challenge took place in November and December 2004. It was sponsored by the Pascal network and was organized mainly by Neil Ireson from University of Sheffield. The aim of the challenge was to assess current Machine Learning methods for Information Extraction.

In chapter 3 we discussed the shortcomings of the methods used to evaluate previous IE systems and the lack of a standard evaluation methodology which would enable us to meaningfully compare the results of different IE systems. The Pascal challenge aimed to define a standard methodology for evaluating IE systems and perform tests of different systems in controlled experiments to determine which aspects of the system contributed to their performance.

The organizers annotated a dataset and specified an evaluation method. All systems were required to use the same basic feature-set. Preprocessing was done using the GATE preprocessor which gives token, POS and orthographic features. It also provided a small number of named entity features (person, location, date).

## 8.2 Pascal Challenge Dataset

The Pascal Challenge dataset [24, 25] consists of 1100 call for papers (CFPs) comprising 850 workshop CFPs and 250 conference CFPs . This corpus divided

<?xml version="1.0" encoding="UTF-8" ?>
<DOC>
<DOCID>1.5-train-3-5-1-CSIT_1999.key</DOCID>
<TEXT>
1st announcement andCALL FOR CONTRIBUTIONS TO

CSIT

The <workshopname>Workshop on Computer Science and Information Technologies</workshopname>

CSIT will be a forum for presentation of new results inresearch, development, and applications in computing and information.The organizers expect both practitioners and theorists to attend.

CSIT will be held from <workshopdate>January 18th to 22nd 1999</workshopdate>, in <workshoplocation>Moscow,Russia</workshoplocation>

Submissions to the workshop should be send by<workshoppapersubmissiondate>September 30</workshoppapersubmissiondate>, in the form of extended abstractsor full papers, to

VEW@na.vuag.lxb

See for full conference details:

http://msu.jurinfor.ru/CSIT/CSIT98CfP-letter.htm

<workshophomepage>http://msu.jurinfor.ru/CSIT/</workshophomepage>

+-+-+-+-+-+-+-+

Institute for Contemporary Education "JurInfoR-MSU",

Vorotnikovsky per, 7

Moscow, 103006 Russia

Phone: +7 (095) 939-1885

Fax: +7 (095) 939-1885

E-mail: vew@lrn.run.of

---
</TEXT>
</DOC>

Figure 8.1: An example Call for Papers

into sub-corpora: training corpus (400 workshop CFPs), test corpus (200 workshop CFPs) and enrich corpus (250 workshop CFPs and 250 conference CFPs). Most of the documents are from the area of computer science and the training and test sets are temporally separate. We used only the train and test corpora.

The annotation process took place over the course of several months. Five versions of the training corpus were released to the participants before the corpus was finalized. Each new version corrected errors that were identified in the previous version.

Figure 8.1 shows an example call for papers. The documents in this dataset tend to be more structured than the other benchmark datasets. There are also strong relationships between most of the fields. For example *workshopacronym* usually occurs just after *workshopname*. The various dates often occur in a certain

| Field | train | test | Examples |
|---|---|---|---|
| *workshopname* | 543 | 245 | Second International Workshop on Time Oriented Business Information Systems |
| *workshopacronym* | 566 | 243 | ZobIS 96, RTSS '98 |
| *workshopdate* | 586 | 326 | June 8-9 2000, September 1st-2nd |
| *workshophomepage* | 367 | 215 | http://www.cs.virginia.edu/wecwis2000 |
| *workshoplocation* | 457 | 224 | Pisa, Italy ; Cottbus ; Cottbus, Germany |
| *workshoppapersubmissiondate* | 590 | 316 | March 1 ; June 3, 1996 |
| *workshopnotificationofacceptencedate* | 391 | 190 | Friday 27th March 1998 |
| *workshopcamerareadycopydate* | 355 | 163 | Mar. 24, 2000 |
| *conferencename* | 204 | 90 | 15th International Conference on Conceptual Modelling |
| *conferenceacronym* | 420 | 187 | ACL/COLING '98, ECAI-2000 |
| *conferencehomepage* | 104 | 75 | www.acm.org/sigs/sigmm/MM99 |

Figure 8.2: Details of the Pascal Challenge dataset

order. e.g. *workshopnotificationofacceptencedate* usually occurs after *workshop-papersubmissiondate*.

Figure 8.2 shows the fields along with examples and the occurrences of each field in the training and test datasets. There are 11 fields: 8 relating to the workshop and 3 relating to the conference. All the fields are single-valued fields. Many of the fields in this dataset are very similar. There are 4 date fields. These often occur together in a document and use the same date format. The dates must be disambiguated using the limited context around each. Similarly, there are two name fields, two acronym fields and two homepage fields, one each for workshop and conference.

## 8.3 Evaluation

We submitted a single entry to the Pascal Challenge. We used a window length of 3 and randomly undersampled 50% of the negative instances. Most systems submitted more than one entry. The full set of results are available at the Pascal Challenge web-site [2]. The web-site states that participants may only reproduce their own results and their rank in comparison to other systems. Table 8.3 shows the rank of our system for each field. When reporting the rank we only count the

| Field | Train | | Test | |
|---|---|---|---|---|
| | f-measure | rank | f-measure | rank |
| *workshopname* | 55.5 | 3 | 43.9 | 5 |
| *workshopacronym* | 68.3 | 4 | 34.7 | 8 |
| *workshopdate* | 70.9 | 5 | 45.7 | 8 |
| *workshophomepage* | 62.8 | 5 | 41.7 | 9 |
| *workshoplocation* | 55.5 | 5 | 38.7 | 8 |
| *workshoppapersubmissiondate* | 70.5 | 6 | 52.7 | 9 |
| *workshopnotificationofacceptancedate* | 71.9 | 7 | 58.8 | 9 |
| *workshopcamerareadycopydate* | 68.7 | 7 | 44.1 | 10 |
| *conferencename* | 66.5 | 2 | 48.3 | 5 |
| *conferenceacronym* | 69.1 | 2 | 20 | 10 |
| *conferencehomepage* | 63.9 | 2 | 5.2 | 10 |

Figure 8.3: ELIE's rank performance on the Pascal Challenge

best performing of each competitor system's entries (some systems submitted several entries with different parameters). On some of the fields our entry is among the top performers for the train set. However our system performs poorly on the date fields. On the test set we perform much worse than we did on the train set.

ELIE performed poorly on the date fields. An examination of the errors revealed that in many cases we identified the date but as the wrong kind of date. Often we would have multiple field predictions for a date and would have identified it as the correct type of date but would have also predicted it as another type of date with higher confidence.

The best performing system on average was Amilcare, submitted by the University of Sheffield group. This system is based on the $(LP)^2$ algorithm. It had exceptional performance on the date and acronym fields while its performance was mediocre on some of the other fields. For example, on the test set it was best for three of the 4 date fields and both the acronym fields. On the *conferenceacronym* field it had an f-measure that was 41% better than the second placed system while for workshop acronym, its recall was 25% better than the second placed system. The reason that $(LP)^2$ outperformed all the other systems is because of its ability to build rules based on the relationship between different fields. There are strong relations between fields in this dataset.

Another system that was one of the top performing was that submitted by ITC-IRST. This was a one-level SVM approach that is similar to ELIE$_{L1}$. It implemented the instance filtering strategy of Gliozzo *et al*.

ELIE performs poorly on this task. There are several reasons for the poor performance. The first is that this dataset has very high imbalance. This means that recall will be low at L1 so there is not much potential for improvement at L2. One of the systems addressed this problem by aggressively filtering instances. Our system does not filter instances in a manner that will address the class imbalance problem. ELIE performs worse on the test data than on the train data. This is also a problem caused by the large imbalance in the dataset. The models learned on the dataset have high precision due to the very high class imbalance. These high precision learners don't transfer well to the test dataset. This also caused overfitting of the training data. Because of the high class imbalance we only make positive predictions for instances that are very close to positive instances from the training data.

The second reason is that ELIE$_{L2}$ learns each field independently. It is not capable of learning contextual information between fields. It deals poorly with the date fields in this dataset. These fields are highly contextual. The occurrence of one of them strongly affects the probability of another occurring. Our system identified most of the dates but it often confuses one type of date with another.

## 8.4 Discussion

The stated aims of the original proposal [1] were:

1. Define a methodology for the fair comparison of machine learning algorithms for IE.

2. Define a publicly available resource for evaluation that will exist beyond the lifetime of the challenge.

3. Perform actual tests of different algorithms in controlled situations so as to understand what works and what doesn't.

The first aim was met by defining a methodology for scoring extractions and defining splits to be used for testing and training the data. The scoring methods that were decided on were the same scoring methods that we chose to use when evaluating our system. No credit is given for partial matches. The system must extract all occurrences of the instances to get full credit (ASO evaluation). This is a conservative and fair method of evaluating an IE system. It is hoped that all future systems will adopt this scoring methodology for evaluation.

The Pascal Challenge resulted in another dataset being made available to the IE community. This dataset is quite different to the other IE benchmark datasets. The documents are longer and there are strong relations between some of the fields in the dataset. This is a valuable resource and will be useful for evaluating IE systems in the future. Unfortunately the annotations for the test data have not been made available to the community - only the annotations for the training set are currently available. This may hamper the adoption of this dataset as another standard benchmark.

The third stated aim of the challenge was to perform controlled tests of the various aspects of the IE system to understand what aspects contribute to performance. The feature-set was fixed for the challenge so that all systems had to use the same feature-set. However this does not allow us to compare the learning algorithms of the other systems. Instead it only excluded one of many variables that can cause performance differences between the systems. Since we don't separate any other aspect of the IE system it merely punishes systems with good feature-sets and benefits systems with poor feature-sets. The actual learning algorithm, the way relational features are handled, the length of window for creating relational features, instance pruning, attribute pruning, post-processing of predictions, etc all contribute to the performance. Fixing the feature-set does not really make for a more valid comparison as there are still so many other variables between systems that can contribute to performance. No other aspects of the IE systems were fixed and the contribution of features to performance was not examined.

$(LP)^2$ was the best performing system on average. A large part of its performance came from its ability to learn rules about relations between fields. This gave it high performance on the data and acronym fields as they had strong relational dependencies on other fields.

## 8.5 Summary

In this chapter we described the Pascal Challenge and discussed ELIE's performance in it.

The Pascal Challenge identified two shortcomings of ELIE:

- It fails when there is very high class imbalance. ELIE does well when the class imbalance is moderate as it takes advantage of high precision models at level one to improve recall at level 2. However when class imbalance is very high, recall at level one becomes so low that there is not much scope for improvement at level 2.
- It does not take advantage of relations between different fields. ELIE assumes that all fields are independent and doesn't use the presence of one field to help learn the presence of another. On the Pascal dataset there is a strong relationship between some of the fields and taking advantage of this information can improve performance.

The Pascal Challenge highlighted the variability in performance of different IE systems. Systems that performed well on some fields performed poorly on others. It also highlighted the effect of data imbalance on performance, the importance of relational information between fields and the need for a wide and varied testbed of IE datasets, each challenging different properties of the IE systems.

# Chapter 9

# Multi-field Contextual Extraction

Level 1 and Level 2 extract a single field at a time. They treat extraction of different fields as independent tasks. However on some tasks there is a strong relationship between the occurrence of different fields. The two-level approach described so far does not exploit this contextual information. On the Pascal dataset, $(LP)^2$'s ability to learn contextual rules based on the occurrence of other fields gave it a big advantage on fields that have a strong contextual dependence on other fields. Our approach of treating all the fields as independent worked on the standard benchmark datasets but failed for the Pascal data. By assuming that all the fields are independent we fail to exploit the structure that exists between the data.

Figure 9.1 shows field-pair probability data for the Seminar Announcements dataset. For field pair (f1, f2) $P_{f2|f1}$ gives the probability of the next field being f2 given that f1 has occurred. If we identify a field as being a *speaker*, the most probable field to follow is *stime* with probability 0.63. The most likely fields to follow *stime* is *etime*. The most likely field to follow *etime* is *location*. This shows that many of the Seminar Announcements have a certain amount of structure with the *speaker* being listed first, the *stime* and *etime* next and the *location* being listed last. It is twice as likely that an *etime* follows an *stime* than an *stime* follows an *etime*. There are some combinations that are very unlikely, e.g. if we identify a *speaker*, the next field is highly unlikely (P=0) to be an *etime*, but very likely to be a *stime* (P=0.69).

Figure 9.2 show the 10 most likely field pairs for the Job Postings dataset. The

| f1 | f2 | $P_{f2\|f1}$ |
|---|---|---|
| *speaker* | *stime* | 0.63 |
| *etime* | *location* | 0.6 |
| *location* | *speaker* | 0.51 |
| *location* | *stime* | 0.47 |
| *stime* | *etime* | 0.44 |
| *stime* | *location* | 0.31 |
| *speaker* | *speaker* | 0.27 |
| *etime* | *stime* | 0.23 |
| *etime* | *speaker* | 0.18 |
| *stime* | *speaker* | 0.15 |
| *stime* | *stime* | 0.1 |
| *speaker* | *location* | 0.1 |
| *location* | *location* | 0.02 |
| *location* | *etime* | 0.01 |
| *etime* | *etime* | 0 |
| *speaker* | *etime* | 0 |

Figure 9.1: Field-pair probabilities for the Seminar Announcements dataset

| $f1$ | $f2$ | $P_{f2\|f1}$ |
|---|---|---|
| *country* | *state* | 0.58 |
| *state* | *city* | 0.55 |
| *recruiter* | *country* | 0.39 |
| *area* | *area* | 0.37 |
| *platform* | *platform* | 0.37 |
| *application* | *application* | 0.36 |
| *language* | *language* | 0.33 |
| *id* | *country* | 0.28 |
| *company* | *city* | 0.24 |
| *company* | *company* | 0.23 |

Figure 9.2: The 10 most likely pair sequences for the Job Postings dataset

*country*, *state* and *city* fields are very likely to co-occur together in that order. If we identify an *area* field, the most likely next field is another *area*. This is also the case for the *platform*, *application* and *language* fields. This is because these are all multi-valued fields that tend to occur in lists in the dataset. E.g. a job application

| f1 | f2 | $P_{f2\|f1}$ |
|---|---|---|
| *status* | *acquired* | 0.58 |
| *sellercode* | *seller* | 0.43 |
| *purchabr* | *acqabr* | 0.43 |
| *acqcode* | *purchaser* | 0.43 |
| *purchaser* | *status* | 0.37 |
| *purchcode* | *purchaser* | 0.36 |
| *acqabr* | *purchabr* | 0.35 |
| *dlramt* | *purchabr* | 0.32 |
| *dlramt* | *acqabr* | 0.29 |
| *purchcode* | *acqabr* | 0.28 |

Figure 9.3: The 10 most likely pair sequences for the Reuters Corporate Acquisitions dataset

may list all languages required in a sequential list. There are many combinations of fields that are very unlikely in this dataset, e.g. *title* is never followed by the *post_date*: the *title* tends to occur near the beginning of documents while the *post_date* tends to occur near the end.

Figure 9.3 show the 10 most likely field pairs for the Reuters Corporate Acquisitions dataset. From this we see that *status* is likely to be followed by *acquired* and *purchaser* is likely to be followed by *status*. This indicates that some sentences will be structured *purchaser - status - acquired*. An example of such a sentence is '<purchaser>General Partners Inc.</purchaser> said it was <status> prepared to raise its bid</status> for <acquired>GenCorp</acquired>.'

Figure 9.4 show the 10 most likely field pairs for the Pascal Challenge dataset. There is quite a bit of structure in this dataset. For example, if we identify a *workshopnotificationofacceptencedate*, it is very likely that the next field will be a *workshopcamerareadycopydate* (P=0.84). This indicates that in the training data, for 84% of occurrences of *workshopnotificationofacceptencedate* the following field is *workshopcamerareadycopydate*. There are many other strong relations in this dataset, many involving the date and acronym fields. The *conferencename* field is most likely to be followed by *conferenceacronym*, *workshopname* is most likely to be followed by *workshopacronym* and *workshoppapersubmissiondate* is very likely to be followed by *workshopnotificationofacceptencedate*.

| f1 | f2 | $P_{f2\|f1}$ |
|---|---|---|
| *workshopnotificationofacceptencedate* | *workshopcamerareadycopydate* | 0.84 |
| *workshoppapersubmissiondate* | *workshopnotificationofacceptencedate* | 0.65 |
| *conferencename* | *conferenceacronym* | 0.5 |
| *workshoplocation* | *workshopdate* | 0.45 |
| *workshopcamerareadycopydate* | *workshopdate* | 0.36 |
| *workshopacronym* | *workshopacronym* | 0.26 |
| *conferencehomepage* | *workshopname* | 0.26 |
| *workshopname* | *workshopacronym* | 0.26 |
| *workshopdate* | *workshoplocation* | 0.26 |
| *workshopname* | *workshopdate* | 0.25 |

Figure 9.4: The 10 most likely pair sequences for the Pascal dataset

The pair-probabilities for these datasets indicate that there in an innate level of structure in the documents and many fields are strongly contextually co-dependent and tend to co-occur together.

However these pair-probabilities do not tell the whole story. They indicate that there is a strong structural relationship between certain fields. However the strength of that relationship also depends on the distance between the fields in the document. In the Pascal dataset the fields are likely to occur very close together in the document whereas in the Seminar Announcements dataset the *location* is likely to follow the *etime* but it is not as likely to occur directly after the *etime*. Thus the relationship between fields is stronger when they are likely to occur close together in a particular sequence than when they are just likely to occur in sequence.

Our system treats the fields as independent. In doing so it fails to take advantage of any structure between fields in the dataset. In this chapter we discuss some enhancements to ELIE that can take account of contextual information between fields.

## 9.1 Adjusting Prediction Confidence

Because we treat the extraction of each field as independent tasks, it is possible for extractions of different fields to overlap or for the same fragment to be extracted

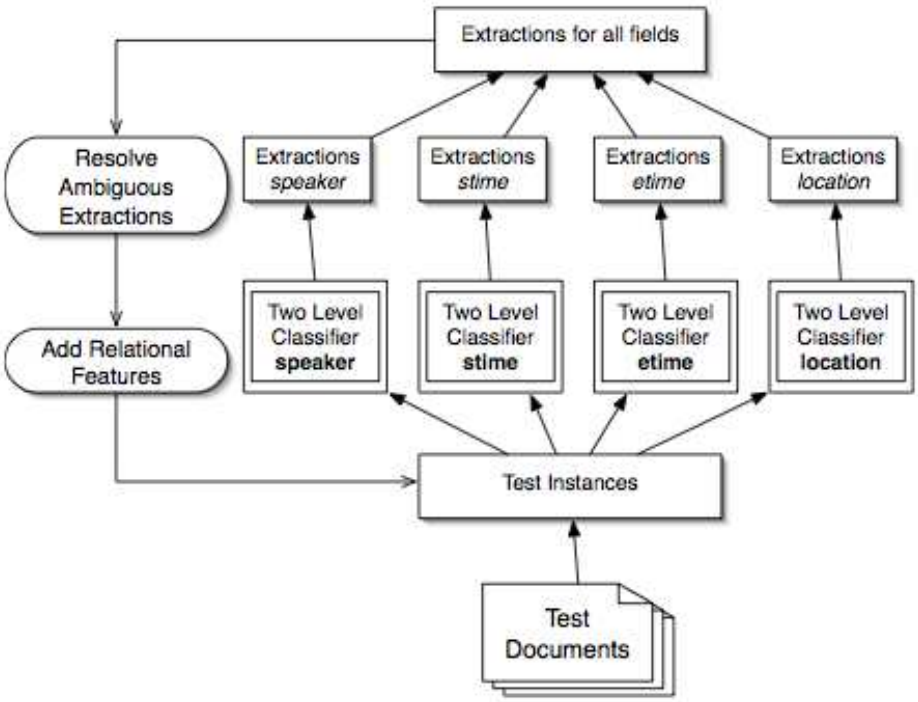


Figure 9.5: Multi-level multi-field extraction

as different fields. This is not generally a problem on the 3 benchmark datasets. However on the Pascal dataset ELIE often extracted multiple date fields for date fragments because the dates are all similar and tend to occur close together in the document. It is not possible for a fragment to be more than one field so it is desirable to eliminate these ambiguous predictions. When we make ambiguous extractions, i.e. we make more than one field prediction for the same piece of text, we must choose one of the predictions and eliminate the others. We choose the extraction with the highest confidence based on the contextual probability. For each ambiguous extraction, we choose the field that is most likely given the previous field.

## 9.2 Adding Mult-Field Contextual Features

In order to take advantage of the structure of the documents and relations between fields, we add multi-field contextual features. For each instance in the dataset,

we add two features: one for the previous field and one for the next field. When learning our models we use the annotations in the dataset to add features to each instance for the next occurring instance and previous occurring instance. When extracting we first run our two-level classifier for each field and use the predictions for these classifiers to add previous instance and next instance features to the dataset. Once these features have been added, we re-run the two-level learner on the dataset with these new features.

Figure 9.5 summarizes this process. In the training phase we learn a two-level model for each field with the addition of features for previous and next fields. When training these features can be taken from the annotations. The extraction process is more complex. From our test documents we generate a set of instances. We pass these instances to the two-level model for each field to get a set of extractions for each fields. We resolve any ambiguous extractions to ensure that different fields don't cover the same text. We then use these extractions to add new features to each instance for previous and next fields. We then pass the dataset with these new features back to each of the two level models and repeat the extraction process with the contextual features present. We could iteratively repeat this process but we only do it once.

## 9.3 Evaluation

Figure 9.6 shows the performance of this multi-level approach on the Seminar Announcements dataset. The addition of the multi-field features gives an increase in performance for the *stime* fields but not any of the others. The f-measure for the *location* and *etime* fields drops when we add the relation features.

## 9.4 Discussion

In this chapter we described a simple approach to extending ELIE to take account of relations between different fields. This approach did not give any significant improvement on the SA dataset. We showed by measuring the pair-probabilities that there is some inherent structure in the dataset but this approach fails to take

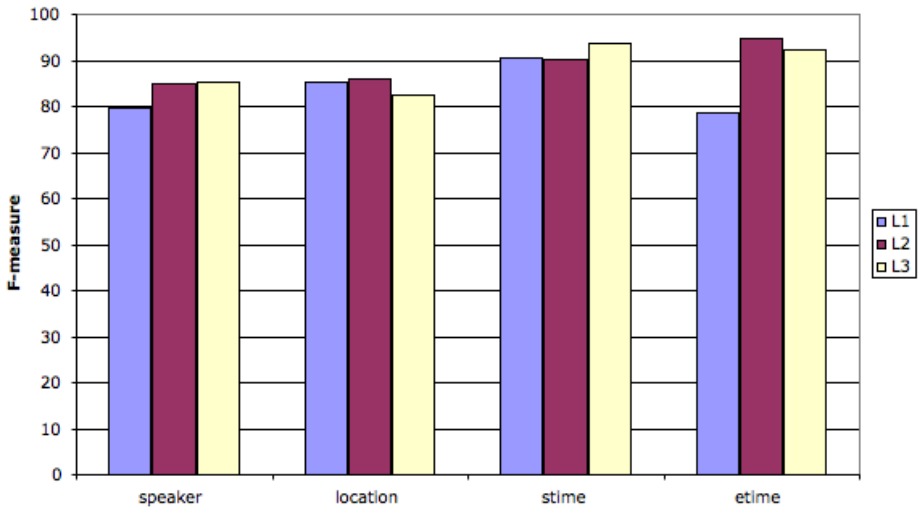


Figure 9.6: F-measure for the Seminar Announcements dataset using multi-field extraction

advantage of this structure.

The features used may be too coarse and adding them to all instances may be adding noise to the data. The features only represent the next field that occurred but don't represent the distance to that field. Fields that occur adjacent to each other are represented in the same way as fields that occur far apart. A more sophisticated approach might break this feature down into several different features that represent how close the field occurred to the previous field or only add the contextual features if a field occurs close to another.

Another problem is that that when training we use the annotations from the training data, so there will be very little noise, whereas for extracting we use the predictions from L2 so there will be lots of noise.

## 9.5 Summary

In this chapter we discuss the benefit of using multi-field contextual information about multiple fields to improve extraction. We motivate this by giving field-pair probabilities for the various datasets. These probabilities show that there is strong contextual dependence between fields in the documents and the ability to use this

information could give large potential for improving performance.

We tried a simple method of adding features for relations between fields and found that it didn't improve performance. A more sophisticated approach is required to take advantage of contextual structure between fields.

# Chapter 10

# Conclusion

In summary, we developed an approach to IE that uses standard Machine Learning techniques. We investigated the various components of this system and analyzed their contribution to the overall performance of the system. We presented a new two-level classification approach to IE that achieves state-of-the-art performance.

## 10.1 Discussion

There are many aspects that contribute to the performance of an IE system. We have investigated the different aspects separately and assessed how important their contribution is to the overall performance of our IE system.

We discussed various methods of evaluating IE systems and the shortcomings of each. We used the most conservative method to evaluate our own system.

Each IE system has different aspects that contribute to its performance. Generally they do not analyze in detail which components contribute to the overall performance of the system. It seems likely that for BWI, boosting contributes strongly to its performance. It seems unlikely that the simple precise rules that it learns would perform as well without the boosting step. Boosting is a general technique that could be applied and incorporated into any of the IE systems described. With $(LP)^2$ it seems likely that a large part of its good performance comes from the contextual rules and the ability to learn rules that use the occurrence of other fields. Learning rules based on other fields is a technique that should be

incorporated into all IE systems to take advantage of the inherent structure in the data and the relationship between fields. With ELIE$_{L1}$ the performance came from a combination of the learning algorithm and the features. We saw that SVM was the best performing learning algorithm and was substantially better than several alternative algorithms. The actual learning algorithm used by (LP)$^2$ is quite simple and is likely that its performance could be improved by using a stronger learning algorithm. The features used also contribute to performance. Our system has a richer feature-set than many of the competitor IE systems. Other IE systems can improve performance by having as rich a feature-set as possible.

The two-level approach to classification that we introduce can give large increases in performance. This method of combining sets of classifiers, and using high-precision classifiers to filter the predictions of high-recall classifiers, can be used by any IE system to improve its performance.

In general, IE systems to date are built from scratch. They combine a variety of components, some of which give good performance and some of which give sub-optimal performance. Future IE systems should combine components that have been shown to have high performance. These components are:

- A rich feature representation that encourages generalization.
- The ability to filter uninformative instances and overcome problems caused by class imbalance.
- A state-of-the-art Machine Learning algorithm that gives high performance on the IE task.
- Enhancements to the basic learning algorithm that improve performance such as boosting, bagging, stacking.
- Methods of combining the predictions of different models to improve performance such as our two-level approach.
- The ability to incorporate information about other fields and represent structure and relations between fields.

## 10.2 Future Work

There are several directions for future work. The first direction is investigating better ways to incorporate contextual information between fields into the model. We showed in chapter 9 that there are strong relations between some of the fields in the dataset. The occurrence of one field can affect the probability of another field occurring. Some fields are likely to occur in sequence and some fields are likely to occur together. ELIE learns to extract fields independently. It doesn't take account of other fields when trying to learn a particular field. The Pascal dataset is one where some fields tend to occur in close proximity to each other. Elie doesn't use this information when learning. Using information about the relationships between different fields and using the predictions of one field to guide predictions of another has the potential to improve performance when the data is structured and fields tend to co-occur in the data.

Our two level approach uses the orphan predictions for starts to identify areas of the documents that we should take a closer look at to see if we should identify an end there. This multi-level approach could be extended from start-end pairs to field-pairs. For example, if we know that *etime* is likely to follow *stime*, we could apply an *etime* classifier that is more specialized at identifying *etime* to areas of documents where we identified an *stime* but didn't identify an *etime*. Another approach might be to alter the confidence of a prediction based on the occurrence of other fields nearby. For example, we could increase the confidence in an *etime* prediction if it occurs soon after an *stime* prediction. Another approach might add features to the dataset that the learner can use to learn the relations between different fields. We tried this approach in chapter 9 but a more detailed representation that takes account of the distance between occurrences of the fields might be more successful.

The second direction for future work is to investigate methods for dealing with the class imbalance problem that occurs when representing IE as a token classification task. ELIE takes advantage of moderate class imbalance. When the class imbalance is moderate, the one-level approach gives high precision and moderate recall. ELIE's two level approach can then improve recall. When the class imbalance is very high, the recall from the one-level approach can be so low that ELIE's

two level approach has little potential for improving recall. The methods that we investigated for undersampling negative instances improved execution time without affecting accuracy but they didn't improve accuracy. We need to find methods that can be applied to datasets with high imbalance that will improve recall to a level where ELIE's two level approach can significantly improve performance.

A third direction for future work is to investigate Active Learning approaches to IE. The annotation of training data is the most time consuming part of IE. It is desirable to minimize the number of documents that need to be annotated while at the same time maximizing performance. Active Learning involves actively selecting which documents are put forward for annotation so as to maximize the information that the learner receives. We would like to select the documents that are most informative and ignore documents that don't give any improvement. One method to select documents could use disagreement among models built with different redundant views of the data. We saw that using only the token features and using all features except the token features both gave good performance. We could build two models using these two different views of the data and choose to annotate documents where they disagree on their predictions.

Another direction for future work is to develop methods for formal theoretical analysis of IE. To date there has been little work in this area and work in IE has been largely empirical. As IE matures it would be desirable to develop formal theoretical models that describe the IE task and can be used to derive performance bounds for IE systems.

# Appendix A

# Publications

2006 **Learning to Classify Documents according to Genre**. Aidan Finn and Nicholas Kushmerick. *Journal of the American Society for Information Science and Technology (JASIST), Special Issue on Computational Analysis of Style, Volume 7, Number 5*.

2005 **Elie: A Two-level Boundary Classification Approach to Adaptive Information Extraction**. Aidan Finn and Nicholas Kushmerick. *First PASCAL Challenges Workshop*.

2005 **Adaptive Information Extraction Research at UCD**. Aidan Finn, Brian McLernon and Nicholas Kushmerick. *Dagstuhl Seminar on Machine Learning for the Semantic Web*.

2004 **Multi-level Boundary Classification for Information Extraction**. Aidan Finn and Nicholas Kushmerick, *European Conference on Machine Learning*.

2004 **Information Extraction by Convergent Boundary Classification**. Aidan Finn and Nicholas Kushmerick, *AAAI-04 Workshop on Adaptive Text Extraction and Mining*.

2003 **Learning to Classify Documents according to Genre**. Aidan Finn and Nicholas Kushmerick. *IJCAI-03 Workshop on Computational Approaches to Style Analysis and Synthesis*.

2003 **Active Learning Selection Strategies for Information Extraction**. Aidan Finn and Nicholas Kushmerick. *ECML-03 Workshop on Adaptive Text Extraction and Mining*.

2002 **Machine Learning for Genre Classification**. Aidan Finn. *Msc. Thesis (University College Dublin)*.

2002 **Genre Classification and Domain Transfer for Information Filtering**. Aidan Finn, Barry Smyth and Nicholas Kushmerick. *European Colloquium on Information Retrieval Research*.

2001 **Fact or Fiction: Content Classification for Digital Libraries**. Aidan Finn, Barry Smyth and Nicholas Kushmerick. *Joint DELOS-NSF Workshop on Personalization and Recommender Systems in Digital Libraries*.

# Appendix B

# A Simple Analysis of Two-level Classifier Behaviour

To date IE research has been largely empirical. Researchers have ideas, they implement them and they are tested on the standard IE datasets. Machine Learning and Text Classification have strong theoretical foundations but to date there has been little formal theoretical analysis of the IE task and the properties of IE systems. Such analysis would be useful in analyzing the behaviour of systems and in estimating upper bounds on their performance.

In this chapter we introduce a simple way to analyze and model ELIE's behaviour.

## B.1 Modelling ELIE's Behaviour

ELIE consists of two levels. Each level consists of start and end classifiers. ELIE's performance depends on the performance of each level and of the start and end classifiers.

Figure B.1 shows the confusion matrices for the L1 and L2 start and end classifiers (TP, FP, TN and FN refer to true positives, false positives, true negatives and false negatives respectively). The performance of the ensemble as a whole depends on the performance of these various components.

We can model the probability of the ensemble correctly predicting a start or

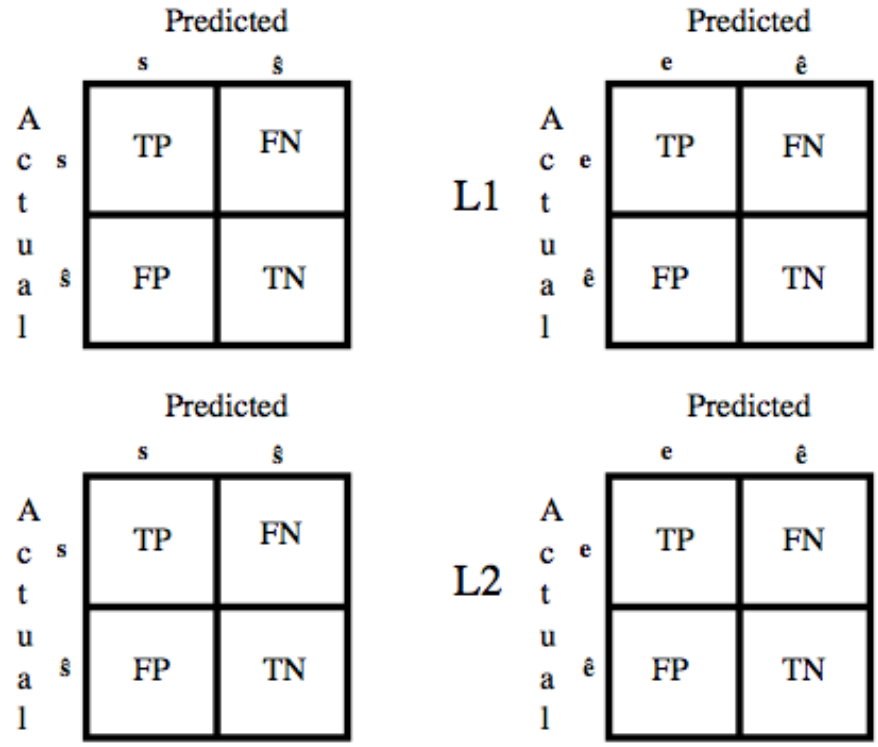


Figure B.1: Confusion matrices for L1 and L2 start and end classifiers

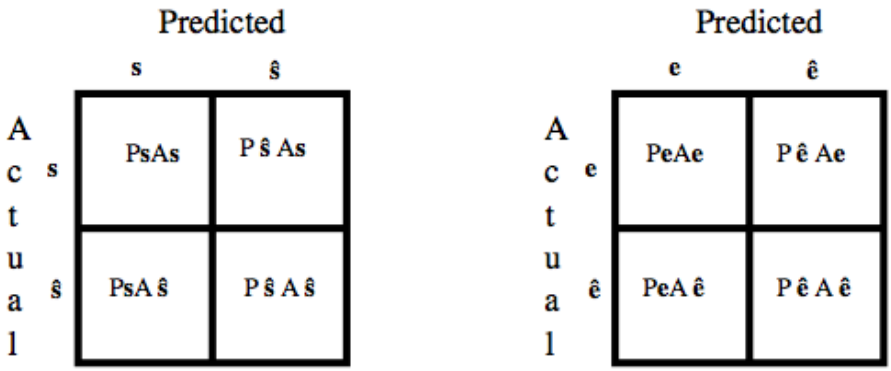


Figure B.2: ELIE ensemble start and end confusion matrices

end using by listing the conditions under which the ensemble will make start or end predictions. Figure B.2 shows the confusion matrices for the ensemble. The notation signifies the probability that a prediction is made in each quadrant of the confusion matrix. For example, $PsAs$ means predict start where it is actually a start. $PsA\hat{s}$ means predict a start where it is actually not a start.

## B.2 Calculating Confusion Matrix Probabilities

We can convert a logical expression such as (AND (OR a b) c) into an expression for computing its probability such as (* (+ p(a) (- p(b) (* p(a) p(b)))) p(c)) where p(x) is the probability of x. To perform this conversion we apply the axioms of probability theory, assuming that everything is independent:

$p(a \vee b) = p(a) + p(b) - p(a \wedge b)$

$p(a \wedge b) = p(a) * p(b)$

$p(\neg a) = 1 - p(a)$

In addition there is mutual exclusion between the four quadrants of the confusion matrix. Each prediction falls into exactly on of the four quadrants for its confusion matrix. If two predictions are mutually exclusive they cannot occur together. Therefore

$p(a \wedge b) = 0 \, if \, a \, and \, b \, are \, mutually \, exclusive$

$p(a \vee b) = p(a) + p(b) \, if \, a \, and \, b \, are \, mutually \, exclusive$

## B.3 ELIE : A Logical Representation

We can describe the conditions under which Elie will make a prediction as logical combinations of the individual classifiers. Listing the conditions under which the ensemble will make a prediction is relatively easy, but we must also represent the fact that there is mutual exclusion between the four quadrants of each confusion matrix. The mutual exclusion for each of the start and end classifiers for level 1 and level 2 are:

```
l1_s_mutex = AND(
     XOR(l1_ps_as,OR(l1_pxs_as,l1_ps_axs,l1_pxs_axs)),
     XOR(l1_pxs_as,OR(l1_ps_as,l1_ps_axs,l1_pxs_axs)),
```

```
    XOR(l1_ps_axs,OR(l1_ps_as,l1_pxs_as,l1_pxs_axs)),
    XOR(l1_pxs_axs,OR(l1_ps_as,l1_ps_axs,l1_pxs_as)));
l1_e_mutex = AND(
    XOR(l1_pe_ae,OR(l1_pxe_ae,l1_pe_axe,l1_pxe_axe)),
    XOR(l1_pxe_ae,OR(l1_pe_ae,l1_pe_axe,l1_pxe_axe)),
    XOR(l1_pe_axe,OR(l1_pe_ae,l1_pxe_ae,l1_pxe_axe)),
    XOR(l1_pxe_axe,OR(l1_pe_ae,l1_pe_axe,l1_pxe_ae)));
l2_s_mutex = AND(
    XOR(l2_ps_as,OR(l2_pxs_as,l2_ps_axs,l2_pxs_axs)),
    XOR(l2_pxs_as,OR(l2_ps_as,l2_ps_axs,l2_pxs_axs)),
    XOR(l2_ps_axs,OR(l2_ps_as,l2_pxs_as,l2_pxs_axs)),
    XOR(l2_pxs_axs,OR(l2_ps_as,l2_ps_axs,l2_pxs_as)));
l2_e_mutex = AND(
    XOR(l2_pe_ae,OR(l2_pxe_ae,l2_pe_axe,l2_pxe_axe)),
    XOR(l2_pxe_ae,OR(l2_pe_ae,l2_pe_axe,l2_pxe_axe)),
    XOR(l2_pe_axe,OR(l2_pe_ae,l2_pxe_ae,l2_pxe_axe)),
    XOR(l2_pxe_axe,OR(l2_pe_ae,l2_pe_axe,l2_pxe_ae)));
mutex = AND(l1_s_mutex, l1_e_mutex, l2_s_mutex, l2_e_mutex);
```

In the above conditions the notation is of the form level_prediction_actual with x signifying negation. So l1_ps_axs indicates that at level 1 we make a start prediction that is actually not a start. The conditions required to make a prediction at L1 are:

```
l1_ps = OR(l1_ps_as, l1_ps_axs);
l1_pe = OR(l1_pe_ae, l1_pe_axe);
```

The conditions required to make a prediction at L2 are:

```
l2_ps = AND(l1_pe, OR(l2_ps_as, l2_ps_axs));
l2_pe = AND(l1_ps, OR(l2_pe_ae, l2_pe_axe));
```

We can estimate the probabilities for each quadrant of the ensemble confusion matrix. The prob function converts a logical expression to an expression for computing its probability.

```
ps_as = OR(l1_ps_as, AND(l1_pe, l2_ps_as));
prob_ps_as = prob(ps_as, mutex);
ps_axs = OR(l1_ps_axs, AND(l1_pe, l2_ps_axs));
prob_ps_axs = prob(ps_axs, mutex);
ps_axs = OR(l1_ps_axs, AND(l1_pe, l2_ps_axs));
```

```
prob_ps_axs = prob(ps_axs, mutex);
pxs_as = OR(l1_pxs_as, AND(l1_pe, l2_pxs_as));
prob_pxs_as = prob(pxs_as, mutex);
pxs_axs = OR(l1_pxs_axs, AND(l1_pe, l2_pxs_axs));
prob_pxs_axs = prob(pxs_axs, mutex);
```

This give us a large complex logical expression that describes the probability for each quadrant of the ensemble confusion matrix. To simplify this expression we can put values at L1 and then use some logical expression software[1] to simplify the resulting expression. If we simplify the expression to a single variable then we can plot the behaviour of the system as a function of that variable.

## B.4 Plotting ELIE's Behaviour

The logical expressions generated that describe ELIE's behaviour are too complex to be useful in practice. However if we make some simplifying assumptions about the values that occur in the confusion matrix we can greatly simplify the resulting logical expressions. If we represent all the values in the quadrants using a single variable, then we can plot the performance of Elie as a function of that variable. We plot the behaviour of the ensemble of start classifiers. The end classifiers will have similar behaviour.

Figure B.3 shows the confusion matrices for L1 and L2 as a function of a single variable. To represent the confusion matrices with a single variable we make some simplifying assumptions.

We assume that the false-positive rate for L1 start and end, and also the false negative rate for L2 start and end, are all equal to some value $\alpha$. We also assume that all remaining probability mass is distributed among the other 3 cells of each confusion matrix (this assumes that the dataset is balanced) and we assume that everything else is probabilistically independent.

This allows us to see if the ensemble is robust to lower precision at level 1 and lower recall at level 2. As $\alpha$ increases, the number of false positives at L1 and the number of false negatives at L2 increases.

[1] We use MuPAD and its 'simplify' function to simplify the logical expressions.

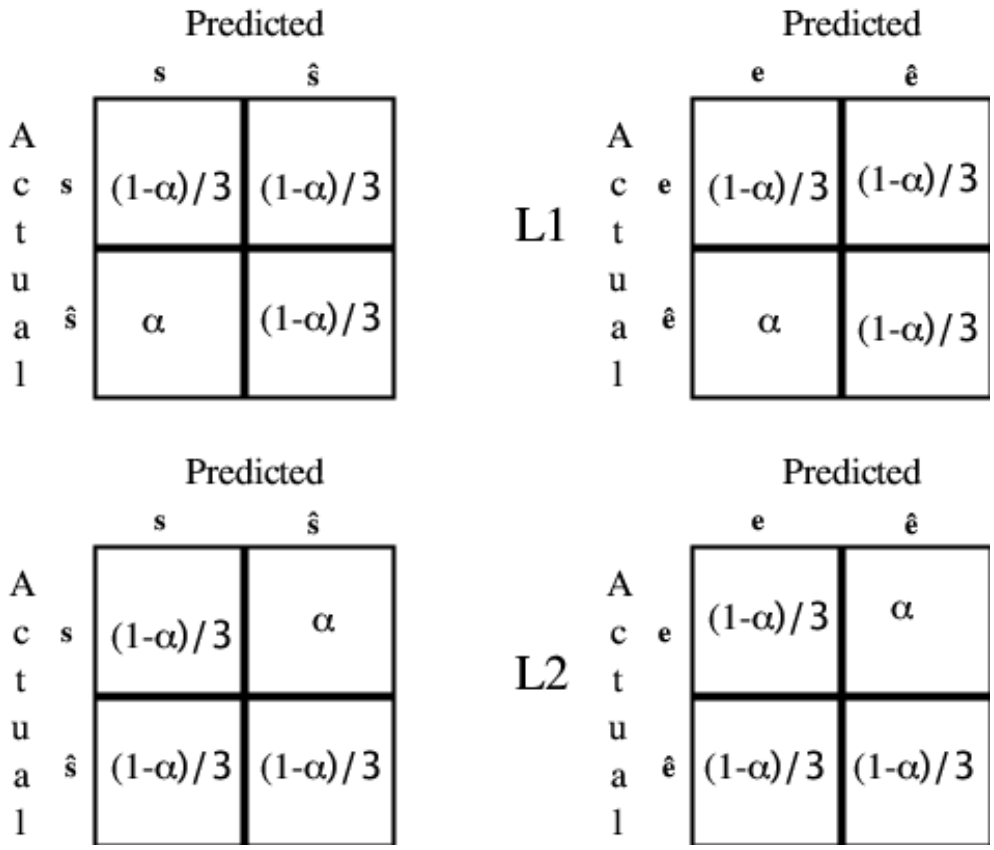


Figure B.3: Confusion matrices for L1 and L2, starts and ends, as a function of a single variable

Figure B.4 shows the probability of predicting a start that is actually a start, i.e. a true positive, as a function of $\alpha$. As $\alpha$ goes to 1, the probability goes to zero. The fall in pPSAS is non-linear and gradual. The upper bound is rather low (approximately 0.4). This comes from the assumption that the remaining probability mass (1-$\alpha$) is distributed equally among the other 3 cells of each confusion matrix. This assumes that the number of false positives and false negatives equal the number of true negatives. Clearly this is not the case as in a real world system (especially if the data is imbalanced) there are likely to be more true negatives than false positives or false negatives. The upper bound would be higher if we made a more optimistic assumption about the distribution of errors.

Figure B.5 shows the probability of predicting a start that is actually not a start,

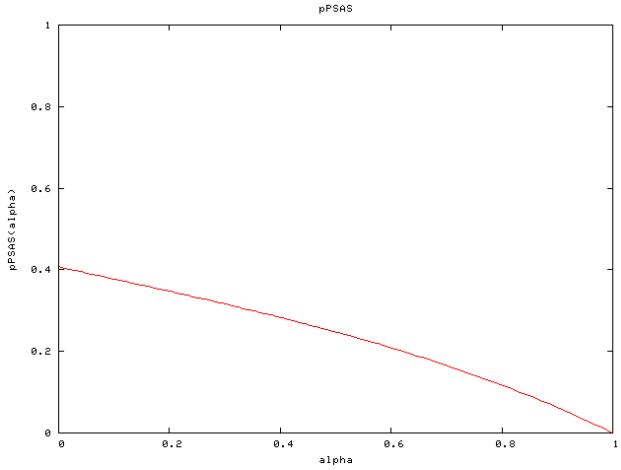


Figure B.4: The probability of predicting a start that is actually a start (PS_AS)

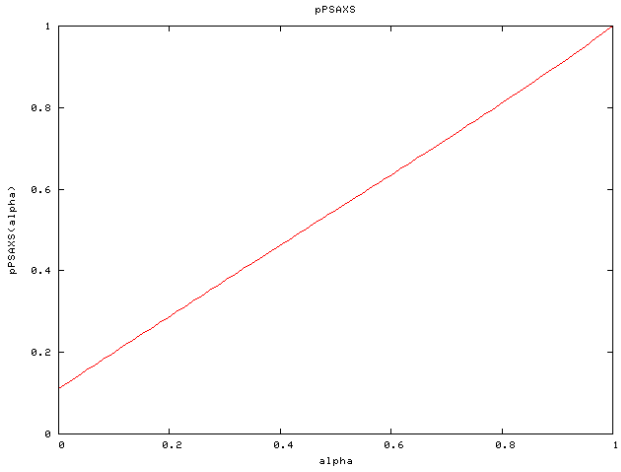


Figure B.5: The probability of predicting a start that is actually not a start (PS_AXS)

i.e. a false positive. This increases constantly as $\alpha$ increases.

Figure B.6 shows the probability of not predicting a start where we should have predicted a start (i.e. false negative). This goes towards 1 as $\alpha$ goes to 1. However it only starts to increase as $\alpha$ goes above 0.5.

Figure B.7 shows the probability of not predicting a start that is actually not a start (true negatives). As $\alpha$ goes to 1, the probability of correctly predicting a

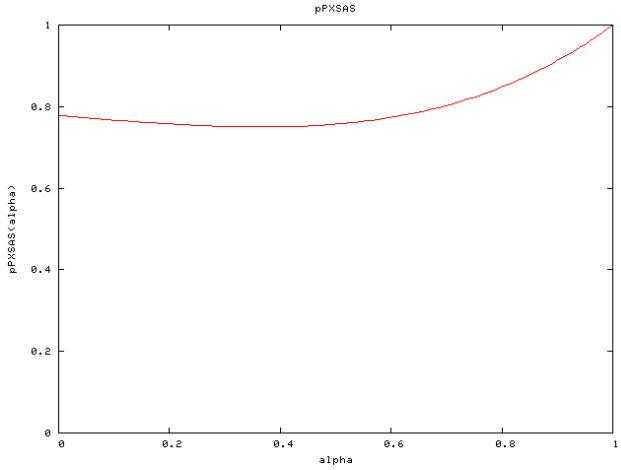


Figure B.6: The probability of not predicting a start that is actually a start (PXS_AS)

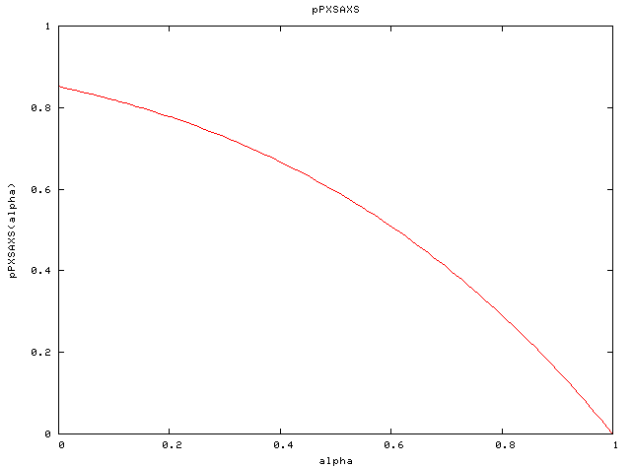


Figure B.7: The probability of not predicting a start that is actually not a start (PXS_AXS)

negative goes to 0.

In both cases where we don't predict a start (figure B.6 and figure B.7) the upper bound is much higher than cases where we do predict a start. This indicates that the ensemble of start classifiers is more likely to predict a token as not being a start than as being a start, i.e. the ensemble is predisposed to negative predictions.

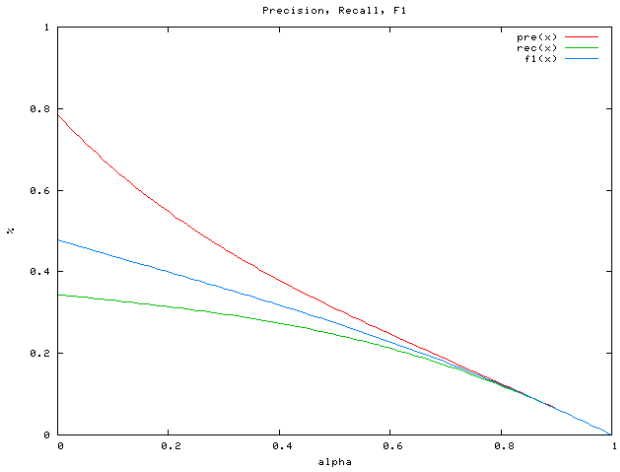


Figure B.8: Precision, Recall and F1 for the ensemble

Figure B.8 shows the precision, recall and f1 for the ensemble of start classifiers based on the probabilities as a function of $\alpha$. It shows that the ensemble is predisposed to high precision. It also shows that as $\alpha$ increases both precision and recall fall. However precision falls at a higher rate than recall. The ensemble's recall is more robust to errors than its precision as it falls more gradually as $\alpha$ increases.

## B.5 Summary

In this chapter we described a method of analyzing the behaviour of ELIE. We listed the logical conditions necessary for the start or end classifiers to make a prediction. We converted these logical expressions to probabilistic expressions and used mathematical analysis software to simplify these expressions. We then made some simplifying assumptions that represented the systems performance as a function of a single variable ($\alpha$). We plotted the expected behaviour of ELIE's start classifier for all four quadrants of the ensemble confusion matrix and used these expected values to plot precision, recall and f-measure.

The gives us a way to model the expected behaviour of our system. We can adjust the distribution $\alpha$ among the cells of the confusion matrix to estimate the performance of the system. The distribution that we chose assumes high precision

at level 1 and high recall at level 2, with the remaining probability mass distributed equally among the remaining cells of the confusion matrix. We saw that the ensemble was predisposed to high precision and lower recall with this configuration and that the ensemble is more likely to predict negatives than positives.

# Appendix C

# Full list of Features

This appendix gives details of all the features used by ELIE. Each instance encodes all these features for the token it is centered on, as well as for a predefined number of tokens before and after.

## C.1 Token features

ELIE uses all tokens that occur in the training data as features. These are dependent on which words occur in the training document and are too numerous to list here.

## C.2 POS features

POS tagging uses Brill's POS tagger. There are 36 POS features.

**CC** Coordinating conjunction. E.g. and, both, but.

**CD** Cardinal number. E.g. mid-1890, nine-thirty.

**DT** Determiner. E.g. all, an, another, any.

**EX** Existential there. E.g. there.

**FW** Foreign word. E.g. alais, je, jour.

**IN** Preposition or subordinating conjunction. E.g. astride, among, upon, whether.

**JJ** Adjective. E.g. regrettable, calamitous, first.

**JJR** Adjective, comparative. E.g. bleaker, braver, breezier, briefer.

**JJS** Adjective, superlative. E.g. calmest, cheapest, choicest, classiest, cleanest.

**LS** List item marker. E.g. A, A., 1.

**MD** Modal. E.g. can, cannot, could, couldn't.

**NN** Noun, singular or mass. E.g. Casino, afghan, shed, thermostat.

**NNS** Noun, plural. E.g. undergraduates, products, bodyguards.

**NP** Proper noun, singular. E.g. Conchita, Trumplane, Christos.

**NPS** Proper noun, plural. Americans, Americas, Amusements.

**PDT** Pre-determiner. E.g. all, both, half, many.

**POS** Possessive ending. E.g. 's

**PP** Personal pronoun. E.g. hers, herself, him, himself.

**PP$** Possessive pronoun. E.g. her, his, mine, my.

**RB** Adverb. E.g. occasionally, unabatingly, maddeningly.

**RBR** Adverb, comparative. E.g. further, gloomier, grander.

**RBS** Adverb, superlative. E.g. best, biggest, bluntest.

**RP** Particle. E.g. aboard, about, across, along.

**Sym** Symbol. E.g. @, =

**To** To.

**UH** Interjection. E.g. Goodbye, Gosh, Wow.

**VB** Verb, base form. E.g. ask, assemble, assess, assign.

**VBD** Verb, past tense. E.g. pleaded, swiped, soaked.

**VBG** Verb, gerund or present participle. E.g. telegraphing, stirring, focusing, angering.

**VBN** Verb, past participle. E.g. chaired, used, experimented.

**VBP** Verb, non-3rd person singular present. E.g. cure, lengthen, brush, terminate.

**VBZ** Verb, 3rd person singular present. E.g. marks, mixes, displeases, seals.

**WDT** Wh-determiner. E.g. that, what, which.

**WP** Wh-pronoun. E.g. what, which, who, whom.

**WP$** Possessive wh-pronoun. E.g. whose.

**WRB** Wh-adverb. E.g. how, however, whenever, where.

## C.3 Gazetteer Features

The gazetteer consists of a set of lists. If a token occurs in one of the lists it is tagged with the name of that list and a feature is added to the instance. The lists that are contained in the Gazetteer are:

**firstname** A list of first-names taken from the U.S. census Bureau.

**lastname** A list of last-names taken from the U.S. census Bureau.

**title** List of titles such as Senator, Miss, Mr, Prof.

**titlepost** Titles that occur after a name e.g. Jr, Esq.

**city** A list of cities from around the world.

**country** A list of countries.

**currencyunit** A list of currencies from around the world.

**location** A list of location descriptors e.g. Creek, County, Valley.

**months** The 12 months of the year.

**numbers** A list of numbers in text form e.g. One, Thirty.

**province** A list of provinces in America and Canada.

**street** A list of street address words. e.g. Avenue, Boulevard.

**timeampm** Words describing whether time is am or pm. e.g. am, noon, midnight.

**timeunit** Units of time, e.g. hours, minutes.

**uspssecondary** A list of secondary location identifiers from the U.S. postal service, e.g. Floor, Room, Apartment.

**states** List of American states.

**stopwords** A list of common stopwords.

## C.4 Orthographic Features

Orthographic features encode information about the type of text that appears in the token. Some of the orthographic features, e.g. symbol, special, punctuation,are defined in the configuration file.

**controlchar** Token is a control character.

**symbol** Token is a symbol. E.g. $, £.

**special** Token is a pre-defined special character. E.g. '\n'.

**punc** Punctuation. E.g. ,.!?

**lbrac** Left bracket. E.g. {[(<.

**rbrac** Right bracket. E.g. }])>.

**word** Token consists of only letters.

**long** Long word. The default length for a long word is 6.

**allupper** Token consists of all upper case characters.

**alllower** Token consists of all lower case characters.

**capitalized** First character of the token is capitalized.

**num** Token is numeric.

***x*digitnum** The number of digits in a numeric token. *x* can take values 1-4. E.g. 1digitnum, 2digitnum.

**snum** Short number. A number consisting of one or two digits.

**schar** Single character.

**lettersanddigits** Token contains both letters and digits.

## C.5 Chunk Features

**NPs** Start of a noun-phrase.

**NPi** Token occurs inside a noun-phrase.

**NPe** End of a noun-phrase.

**VPs** Start of a verb-phrase.

**VPi** Token occurs inside a noun-phrase.

**VPe** End of a verb-phrase.

## C.6 ERC Features

**person** Matches gazetteer sequences that are patterns for a person name. E.g. title-firstname-lastname

**time** Matches gazetteer and orthographic sequences that are patterns for times. E.g. 1digitnum-punc-2digitnum-ampm

## C.7 Pair features

We create a pair feature for each pair of features in the POS, gazetteer, orthographic, chunk and ERC features. E.g. Token is firstname and capitalized.

# Appendix D

# Informative Features

This chapter lists informative features for the start and end classifiers of each field in the three benchmark datasets. For each field, we list the ten most informative features, as ranked by Information Gain, for a single run of ELIE using a 50:50 split. So for each field, the features listed were the 10 most informative features chosen by the model based on 50% of the training data. This gives us an idea of which features and which kinds of features are informative for the various fields.

The features are formatted as TYPE_VALUE_POSITION. TYPE is one of Tok (Tok), G (gazetteer), T (orthographic), POS (part-of-speech), C (chunk) or E (entities). POSITION can have value -4 to 4 indication the occurrence of the feature in relation to the current Tok. For example, G_personfirst_-1 indicates that the token before the current one was tagged as a first-name by the gazetteer.

## D.1 Seminar Announcements

| | *speaker* | | *stime* | |
|---|---|---|---|---|
| | start | end | start | end |
| 1 | E_person_1 | G_personlast_0 | T_1digitnum_0 | E_time_-1 |
| 2 | E_person_0 | E_person_-1 | T_2digitnum_2 | E_time_0 |
| 3 | G_personfirst_0 | E_person_0 | T_snum_0 | E_time_-2 |
| 4 | G_personlast_1 | G_personfirst_-1 | E_time_1 | G_ampm_0 |
| 5 | POS_NNP_0 | POS_NNP_0 | E_time_0 | G_time_0 |
| 6 | T_capitalized_0 | T_capitalized_0 | T_num_0 | Tok_pm_0 |
| 7 | C_NPs_0 | C_NPe_0 | E_time_2 | Tok_:_3 |
| 8 | C_NPe_0 | POS_NNP_-1 | Tok_:_1 | Tok_:_1 |
| 9 | POS_NNP_1 | C_NPs_-1 | Tok_time_-2 | E_time_-3 |
| 10 | T_capitalized_1 | T_capitalized_-1 | T_snum_2 | T_snum_-3 |

| | *etime* | | *location* | |
|---|---|---|---|---|
| | start | end | start | end |
| 1 | Tok_-_-1 | Tok_-_-4 | Tok_place_-2 | T_4digitnum_0 |
| 2 | T_1digitnum_0 | G_time_0 | Tok_hall_1 | POS_NNP_-1 |
| 3 | T_2digitnum_2 | G_ampm_0 | POS_NNP_0 | T_captialized_-1 |
| 4 | E_time_0 | Tok_pm_0 | E_time_-4 | Tok_\n_1 |
| 5 | E_time_1 | E_time_-1 | Tok_wean_0 | T_special_1 |
| 6 | G_ampm_3 | E_time_-2 | T_4digitnum_1 | Tok_hall_0 |
| 7 | G_time_3 | E_time_0 | G_time_-4 | T_num_0 |
| 8 | Tok_pm_3 | T_1digitnum_-3 | G_ampm_-4 | Tok_weh_-1 |
| 9 | T_snum_0 | Tok_00_-1 | Tok_weh_0 | Tok_hall_-1 |
| 10 | Tok_:_1 | E_time_-3 | Tok_:_-1 | Tok_5409_0 |

## D.2 Job Postings

| | *id* | | *title* | |
|---|---|---|---|---|
| | start | end | start | end |
| 1 | Tok_<_-1 | Tok_>_1 | POS_NNP_0 | Tok_programmer_0 |
| 2 | Tok_message_-3 | T_rbrac_1 | T_capitalized_0 | C_NPe_0 |
| 3 | T_lbrac_-1 | Tok_:_4 | T_long_0 | POS_NNP_-1 |
| 4 | Tok_:_-2 | Tok_reply-to_3 | Tok_programmer_0 | T_long_0 |
| 5 | T_special_-4 | Tok_nntp_3 | C_NPs_0 | POS_NNP_0 |
| 6 | Tok_\n_-4 | Tok_\n_2 | Tok_austin_-2 | T_capitalized_0 |
| 7 | POS_NNP_-3 | T_special_2 | T_word_0 | Tok_engineer_0 |
| 8 | T_num_0 | Tok_com_0 | Tok_title_-2 | Tok_developer_0 |
| 9 | T_num_2 | T_capitalized_3 | G_city:-2 | T_capitalized_-1 |
| 10 | T_punc_-1 | POS_NNP_3 | POS_NNP_1 | C_NPs_-1 |

| | *company* | | *salary* | |
|---|---|---|---|---|
| | start | end | start | end |
| 1 | POS_NNP_0 | Tok_victina_0 | Tok_$_1 | Tok_$_-2 |
| 2 | Tok_victina_0 | POS_NNP_0 | Tok_$_0 | Tok_k_0 |
| 3 | T_allupper_0 | T_allupper_0 | Tok_to_0 | Tok_$_-1 |
| 4 | Tok_ctg_0 | POS_VBZ_1 | POS_TO_0 | Tok_to_-3 |
| 5 | Tok_systems_1 | Tok_ctg_0 | T_symbol_1 | POS_TO_-3 |
| 6 | Tok_\n_-1 | Tok_systems_1 | Tok_k_3 | T_symbol_-2 |
| 7 | T_special_-1 | Tok_alliance_0 | T_symbol_1 | T_2digitnum_-1 |
| 8 | Tok_alliance_0 | Tok_is_1 | T_2digitnum_2 | Tok_$_-3 |
| 9 | T_special_-2 | Tok_international_2 | Tok_up_-1 | Tok_000_0 |
| 10 | Tok_\n_-2 | Tok_\n_-2 | Tok_$_4 | Tok_sat_4 |

| | *recruiter* | | *state* | |
|---|---|---|---|---|
| | start | end | start | end |
| 1 | Tok_resource_0 | Tok_resource_-1 | Tok_tx_0 | Tok_tx_0 |
| 2 | POS_NNP_0 | Tok_spectrum_0 | G_location_-2 | G_location_-2 |
| 3 | Tok_spectrum_1 | POS_NNP_-1 | T_allupper_0 | T_allupper_0 |
| 4 | C_NPs_0 | POS_NNP_0 | Tok_-_-1 | Tok_-_1 |
| 5 | C_NPe_1 | T_capitalized_-1 | POS_NNP_0 | POS_NNP_0 |
| 6 | T_capitalized_0 | Tok_quorum_3 | G_country_-2 | G_country_-2 |
| 7 | POS_NNP_1 | Tok_5050_2 | Tok_austin_2 | Tok_austin_2 |
| 8 | Tok_quorum_4 | Tok_dr_4 | Tok_us_-2 | Tok_us_-2 |
| 9 | Tok_5050_3 | C_NPs_-1 | G_city_2 | G_city_2 |
| 10 | T_capitalized_1 | C_NPe_0 | Tok_-_-1 | Tok_-_-1 |

| | *city* | | *country* | |
|---|---|---|---|---|
| | start | end | start | end |
| 1 | G_city_0 | G_city_0 | G_country_0 | G_country_0 |
| 2 | Tok_austin_0 | Tok_austin_0 | Tok_us_0 | Tok_us_0 |
| 3 | G_location_0 | G_personfirst_0 | G_location_0 | G_location_0 |
| 4 | G_personfirst_0 | G_location_0 | POS_PRP_0 | POS_PRP_0 |
| 5 | POS_NNP_0 | POS_NNP_0 | T_allupper_0 | T_allupper_0 |
| 6 | T_capitalized_0 | T_capitalized_0 | Tok_tx_2 | Tok_tx_2 |
| 7 | Tok_tx_-2 | Tok_tx_-2 | Tok_-_1 | Tok_-_1 |
| 8 | T_long_0 | T_long_0 | Tok_-_3 | Tok_-_3 |
| 9 | Tok_-_-1 | Tok_-_-1 | Tok_austin_4 | Tok_austin_4 |
| 10 | Tok_us_-4 | Tok_us_-4 | G_city_4 | G_city_4 |

| | *language* | | *platform* | |
|---|---|---|---|---|
| | start | end | start | end |
| 1 | Tok_c_0 | Tok_c_-2 | POS_NNP_0 | POS_NNP_0 |
| 2 | T_allapper_0 | Tok_+_0 | Tok_windows_0 | Tok_nt_0 |
| 3 | Tok_+_2 | Tok_+-1 | Tok_unix_0 | Tok_unix_0 |
| 4 | Tok_+_1 | Tok_allupper_0 | T_allupper_0 | T_allupper_0 |
| 5 | T_word_0 | Tok_c_0 | Tok_nt_0 | Tok_windows_-1 |
| 6 | Tok_cobol_0 | Tok_,_1 | T_word_0 | Tok_windows_0 |
| 7 | Tok_visual_0 | POS_NNP_0 | T_capitalized_0 | Tok_95_0 |
| 8 | POS_NNP_0 | T_allupper_-2 | T_alllower_0 | T_punc_0 |
| 9 | Tok_+_3 | Tok_cobol_0 | T_punc_0 | T_alllower_0 |
| 10 | Tok_/_-1 | T_symbol_-1 | Tok_95_1 | T_schar_0 |

| | *application* | | *area* | |
|---|---|---|---|---|
| | start | end | start | end |
| 1 | POS_NNP_0 | POS_NNP_0 | POS_NNP_0 | POS_NNP_0 |
| 2 | Tok_oracle_0 | Tok_oracle_0 | Tok_,_-1 | T_allupper_0 |
| 3 | T_allupper_0 | T_capitalized_0 | T_allupper_0 | Tok_,_1 |
| 4 | T_word_0 | Tok_db_-1 | T_punc_0 | T_punc_0 |
| 5 | T_capitalized_0 | T_long_0 | T_word_0 | T_word_0 |
| 6 | Tok_db_0 | Tok_sysbase_0 | Tok_,_1 | Tok_,_-1 |
| 7 | T_long_0 | T_punc_0 | T_schar_0 | Tok_mfc_0 |
| 8 | Tok_sysbase_0 | Tok_,_1 | C_NPs_0 | T_schar_0 |
| 9 | Tok_,_-1 | T_alllower_1 | Tok_mfc_0 | T_long_0 |
| 10 | T_schar_0 | T_word_0 | C_NPe_1 | Tok_client/server_0 |

| | *required years exp.* | | *desired years exp.* | |
|---|---|---|---|---|
| | start | end | start | end |
| 1 | T_1digitnum_0 | Tok_years_1 | Tok_years_1 | Tok_years_1 |
| 2 | T_snum_0 | T_1digitnum_0 | T_snum_0 | T_1digitnum_-2 |
| 3 | T_num_0 | T_snum_0 | Tok_-_-1 | T_snum_0 |
| 4 | Tok_years_1 | POS_NNS_1 | T_1digitnum_0 | Tok_-_-1 |
| 5 | Tok_3_0 | T_num_0 | T_1digitnum_-2 | T_1digitnum_0 |
| 6 | Tok_years_2 | Tok_+_0 | T_num_0 | Tok_5_0 |
| 7 | Tok_+_1 | T_schar_0 | Tok_5_0 | T_num_0 |
| 8 | Tok_2_0 | Tok_3_0 | T_snum_-2 | POS_NNS_1 |
| 9 | T_schar_0 | T_1digitnum_-1 | POS_VBP_2 | T_snum_-2 |
| 10 | T_symbol_1 | Tok_least_-1 | Tok_experience_2 | POS_VPB_2 |

| | *required degree* | | *desired degree* | |
|---|---|---|---|---|
| | start | end | start | end |
| 1 | T_allupper_0 | Tok_degree_1 | Tok_msme_0 | Tok_b_-3 |
| 2 | POS_NNP_0 | T_allupper_0 | Tok_bsme_-2 | Tok_bsme_-2 |
| 3 | Tok_bs_0 | Tok_bscs_0 | T_allupper_0 | Tok_msms_-0 |
| 4 | Tok_bscs_0 | Tok_bs_0 | Tok_science_3 | Tok_science_3 |
| 5 | Tok_qualifications_-4 | Tok_qualifications_-4 | Tok_preferred_-2 | Tok_associates_2 |
| 6 | Tok_bachelor_0 | POS_NNP_0 | Tok_or_4 | Tok_degree_3 |
| 7 | Tok_s_2 | Tok_b_-3 | Tok_b_0 | Tok_computer_2 |
| 8 | Tok_degree_1 | Tok_science_4 | Tok_computer_2 | Tok_masters_0 |
| 9 | Tok_b_0 | Tok_ba_0 | T_allupper_-2 | Tok_s_-1 |
| 10 | T_word_0 | Tok_bsee_0 | POS_NNP_0 | POS_NNPS_2 |

| | *post date* | |
|---|---|---|
| | start | end |
| 1 | G_month_1 | G_date_-1 |
| 2 | G_date_1 | G_month_-1 |
| 3 | Tok_1997_2 | Tok_1997_0 |
| 4 | E_time_3 | E_time_4 |
| 5 | E_time_4 | E_time_3 |
| 6 | Tok_sep_1 | E_time_2 |
| 7 | T_4digitnum_2 | E_time_1 |
| 8 | T_2digitnum_3 | Tok_sep_-1 |
| 9 | T_snum_0 | T_4digitnum_0 |
| 10 | T_snum_3 | T_2digitnum_3 |

## D.3 Reuters Corporate Acquisitions

| | *acquired* | | *purchaser* | |
|---|---|---|---|---|
| | start | end | start | end |
| 1 | POS_NNP_0 | T_capitalized_0 | Tok_-_-1 | T_capitalized_0 |
| 2 | T_capitalized_0 | POS_NNP_0 | T_symbol_-1 | POS_NNP_0 |
| 3 | C_NPs_0 | Tok_inc_0 | POS_NNP_0 | T_capitalized_-1 |
| 4 | T_capitalized_1 | T_capitalized_-1 | G_date_-3 | POS_NNP_-1 |
| 5 | C_NPi_1 | C_NPe_0 | G_month_-3 | C_NPe_0 |
| 6 | POS_NNP_1 | POS_NNP_-1 | T_capitalized_0 | Tok_inc_0 |
| 7 | T_alllower_0 | C_NPi_-1 | C_NPs_0 | Tok_said_1 |
| 8 | T_capitalized_2 | Tok_corp_0 | T_capitalized_1 | Tok_corp_0 |
| 9 | T_alllower_1 | T_capitalized_-2 | T_snum_-2 | T_alllower_0 |
| 10 | POS_NNP_2 | POS_NNP_-2 | Tok_march_-3 | Tok_it_2 |

| | *seller* | | *acqabr* | |
|---|---|---|---|---|
| | start | end | start | end |
| 1 | POS_NNP_0 | POS_NNP_0 | POS_NNP_0 | POS_NNP_0 |
| 2 | Tok_-_-1 | T_capitalized_0 | T_capitalized_0 | T_capitalized_0 |
| 3 | T_symbol_-1 | T_capitalized_-1 | T_alllower_0 | T_alllower_0 |
| 4 | G_month_-3 | POS_NNP_-1 | G_stopword_0 | G_stopword_0 |
| 5 | G_date_-3 | C_NPe_0 | T_allupper_0 | T_allupper_0 |
| 6 | T_capitalized_0 | Tok_inc_0 | Tok_’_1 | Tok_’_1 |
| 7 | C_NPs_0 | Tok_corp_0 | T_long_0 | T_long_0 |
| 8 | T_capitalized_1 | Tok_said_1 | POS_IN_-1 | T_word_0 |
| 9 | POS_NNP_1 | T_alllower_0 | T_word_0 | Tok_s_2 |
| 10 | T_snum_-2 | Tok_it_2 | T_schar_0 | T_schar_0 |

| | *purchabr* | | *sellerabr* | |
|---|---|---|---|---|
| | start | end | start | end |
| 1 | POS_NNP_0 | POS_NNP_0 | POS_NNP_0 | POS_NNP_0 |
| 2 | T_alllower_0 | T_alllower_0 | T_alllower_0 | T_alllower_0 |
| 3 | Tok_reuter_-1 | T_capitalized_0 | Tok_reuter_-1 | T_allupper_0 |
| 4 | T_capitalized_0 | T_allupper_0 | T_allupper_0 | Tok_{_1 |
| 5 | T_allupper_0 | G_stopword_0 | G_personlast_-1 | T_lbrac_1 |
| 6 | G_personlast_-1 | Tok_{_1 | T_capitalized_0 | Tok_}_3 |
| 7 | G_stopword_0 | Tok_reuter_-1 | Tok_._-3 | T_rbrac_3 |
| 8 | Tok_._-3 | T_lbrac_1 | Tok_}_3 | T_capitalized_0 |
| 9 | T_word_0 | POS_VBD_1 | Tok_{_1 | G_stopword_0 |
| 10 | T_word_-2 | Tok_}_3 | T_rbrac_3 | Tok_reuter_-1 |

| | *acqloc* | | *dlramt* | |
|---|---|---|---|---|
| | start | end | start | end |
| 1 | T_capitalized_0 | G_location_0 | T_num_0 | Tok_dlrs_0 |
| 2 | G_location_0 | T_capitalized_0 | Tok_mln_1 | Tok_mln_-1 |
| 3 | POS_NNP_0 | G_province_0 | Tok_undisclosed_0 | POS_NNS_0 |
| 4 | POS_IN_-1 | T_alllower_0 | T_snum_0 | T_num_-2 |
| 5 | Tok_in_-1 | POS_NNP_0 | POS_IN_-1 | Tok_disclosed_0 |
| 6 | G_city_0 | G_city_0 | Tok_for_-1 | C_NPs_-1 |
| 7 | T_alllower_0 | Tok_,_1 | Tok_dlrs_2 | Tok_._1 |
| 8 | T_long_0 | T_punc_1 | T_3digitnum_0 | C_NPe_0 |
| 9 | G_stopword_-1 | G_stopword_0 | G_stopword_-1 | Tok_undisclosed_0 |
| 10 | G_province_0 | Tok_based_1 | Tok_about_-1 | Tok_\n_2 |

| | *status* | | *acqbus* | |
|---|---|---|---|---|
| | start | end | start | end |
| 1 | POS_VBN_0 | T_alllower_0 | POS_NN_0 | T_long_0 |
| 2 | Tok_agreed_0 | T_long_0 | T_long_0 | POS_NNS_0 |
| 3 | T_long_0 | Tok_completed_0 | G_stopword_0 | POS_NN_0 |
| 4 | T_alllower_0 | Tok_agreed_-2 | T_alllower_0 | C_NPe_0 |
| 5 | Tok_it_-2 | POS_VBN_0 | Tok_oil_0 | T_alllower_0 |
| 6 | Tok_completed_0 | C_VPe_0 | POS_NN_1 | G_stopword_0 |
| 7 | POS_PRP_-2 | G_stopword_0 | C_NPs_0 | T_word_0 |
| 8 | Tok_has_-1 | Tok_acquired_0 | T_word_0 | C_NPi_-1 |
| 9 | Tok_it_-1 | Tok_principle_0 | Tok_and_1 | Tok_products_0 |
| 10 | POS_VBD_0 | Tok_agreement_0 | C_NPi_1 | POS_NN_-1 |

# Appendix E

# Using ELIE

ELIE is a tool for adaptive information extraction. It also provides a number of other text processing tools e.g. POS tagging, chunking, gazetteer, stemming. It is written in Python.

## E.1 Installation

Requirements:

- Python 2.1 or higher
- Java 2 or higher
- Weka (included in distribution)
- Brilltag (if you intend to use datasets other than those provided)

Unzip the ELIE archive. Edit the *basedir, BRILLTAGPATH* and *java* variables in the file *config.py* to describe your own system. Add *$ELIEHOME/lib/weka.jar* to your java classpath.

## E.2 Usage

ELIE contains the following executable files

**evaluation.py** The main way to run ELIE.

**scorer.py** Calculate performance measures from ELIE logs.

**extractor.py** Performs basic learning and extraction.

**preprocessCorpus.py** Preprocesses a corpus of text files.

**tagging.py** Does POS, chunking etc. on a text file.

You can execute these files without any arguments to get usage information.

### E.2.1 Input Format

Documents should be stored in text files with one document per text-file. Fields should be marked using the syntax <*field*> ... </*field*>.

### E.2.2 Preprocessing

This stage adds tokenization, orthographic, POS, chunking and gazetteer information to the input files and stores it using ELIE's own format. This stage only needs to be done once for each document collection. Running

```
preprocessCorpus.py datasetDirectory
```

will create a new directory called *datasetDirectory.preprocessed* which contains all the files in ELIE's internal format. Note the input files shouldn't contain any unusual control characters and for every <field> there must be a corresponding </field>.

### E.2.3 Running ELIE

The recommended way to run ELIE is using the file *evaluation.py*. It takes the following parameters.

```
-f field
```

A list of the fields to be extracted surrounded by quotes e.g. "speaker stime etime location"

```
-t trainCorpusDirectory
```

The directory that contains the pre-processed corpus.

```
-D dataDirectory
```

The directory to save ELIE's output and temporary files in.

```
[-T testCorpusDirectory]
```

Optionally specify a directory that contains the pre-processed test corpus. If no test corpus is specified ELIE will do a random split of the training corpus.

```
[-s splitfilebase]
```

Specify a set of pre-defined splits for the training data.

If -t and -T are are set, then ELIE will train on trainCorpusDirectory and test on testCorpusDirectory. Otherwise it will do repeated random splits on trainCorpusDirectory. Other options include:

```
-p set train proportion
```

For a random split experiment set the proportion of the data to use for training. The default value is 0.5.

```
-n number of trials
```

For a random split experiment set the number of trials. The default value is 10.

```
-v version info
-h help
```

The corpora directories should contain preprocessed files only i.e. those created by preprocessCorpus.py. The dataDirectory is where ELIE will store all its intermediate and output files. The splitfilebase argument can used be for predefined splits.

## E.3 Output

The detail of ELIE's printed output is controlled using the parameter *config.verbosity*.

ELIE produces several logfiles that can be used by the bwi-scorer or ELIE's own scorer (scorer.py). The logfile names have form *name.field.elie.number.level.log*.

The split files name has the form *elie.field.number.split*. Each split-file lists the name of each training file, one per line, followed by a separator, followed by the name of each test file, one per line.

These are located in the specified dataDirectory. For a random split experiment ELIE will produce a split file for each iteration. Each split file lists the files used for training and testing. To use pre-defined splits, pass the base of the splitfiles using the -s option.

## E.4 Configuration

The file *config.py* contains all the configuration options. In this section we describe these parameters and their default values.

The config.py file contains several constants that ELIE uses.

```
basedir = '/home/aidan/IE/Elie5'
```

This is the full path to the directory where ELIE is installed

```
BRILLTAGPATH='/usr/Brilltag/Bin_and_Data/tagger'
```

This is the full path to the Brilltag tagger binary.

```
verbosity = 2
```

This controls the level of output that ELIE produces. Higher numbers produce more output. It takes values 0 to 5.

```
java = 'java -mx1900000000 -oss1900000000 '
```

This is the command to call the java runtime. You can add any java parameters here. It is a good idea to allocate plenty of memory to the java interpreter.

```
use_psyco = 0
```

This can have values 0 or 1. Psyco is a program for dynamically compiling python scripts for improved execution time. Enabling psyco will make ELIE run faster but will use a lot more memory. On large experiments this doesn't give much improvement as most of the time is spent inside WEKA.

```
learner = 'SMO'
```

This setting controls which learning algorithm is used. SMO is the default. Available options are: 'knn', 'm5', 'kstar', 'hyper', 'm5rules', 'j48', 'OneR', 'neural', 'winnow', 'LMT', 'jrip', 'SMO', 'prism', 'PART', 'ridor', 'bayes'.

The punctuation, symbols, lbrackets, rbrackets, quotes, longword, usable_tags, reserved_characters and special_tokens parameters are constants that control the behavior of the tokenizer and preprocessor. In general they shouldn't be changed.

The following options control ELIE's behavior. These are the only options that the user needs to change after installation.

```
window = 4
```

This controls the number of tokens for which relation information before and after the current token is encoded.

```
m_window = 10
```

This controls the length of the L2 window: How many instances before and end and after a start to use for training.

```
stem = 0
suffix = 0
```

Whether to use the token stems and token suffixes as features.

```
token = 1
pos = 1
types = 1
gaz = 1
chunk = 1
erc = 1
```

These control which feature-sets to use. Set a value to 0 to disable using those features.

```
filter_n_attributes = 5000
```

This controls how many attributes to use for learning. We can set it to use the top *n* features as ranked by Information Gain.

```
filter_threshold = 0
```

We can set a threshold here for attribute filtering. E.g. setting this to 0.1 would mean that we use the top 10% of attributes as ranked by Information Gain.

```
undersample = 0
```

This controls whether to use random undersampling of instances. Setting it to 0.8 would randomly delete 80% of the negative instances

```
prune_instances = 0
```

This controls whether to prune uninformative instances. Setting it to 80 would prune 80% of the instances as ranked by the informativeness of the word token.

## E.5 Examples

ELIE takes input documents that are in its own format. This format adds the gazetteer, POS, orthographic features etc. To translate a corpus into this format we use the preprocessCorpus.py command.

```
preprocessCorpus.py ./train
```

This creates a new directory called ./train.preprocessed which contains processed versions of all the files that were in ./train. This only needs to be done once per corpus.

```
evaluation.py -t ./train.preprocessed -T ./test.preprocessed
     -D ./tmp -f 'speaker stime etime location'
```

This command does a single train-test run using the files in train.preprocessed for training and the files in test.preprocessed for testing. The log files are stored in ./tmp. Four fields are extracted: speaker, stime, etime, location.

```
evaluation.py -t ./train.preprocessed -D ./results
     -n 1 -p 0.8 -f 'speaker stime etime location'
```

This does a single random test/train split. The files in train.preprocessed are randomly assigned to the train or test set with 80% of them assigned to the train set and 20% to the test set. The log files and the split files are stored in ./results

```
evaluation.py -t ./train.preprocessed -D ./tmp
     -s ./tmp/elie.speaker. -f 'speaker stime etime location'
```

In this example we use the -s option to tell Elie to use predefined train-test splits. The split files define which files from ./train.preprocessed are allocated to the train and test sets. The -s option takes the base of the splitfile name. Splitfile names end in .split and should be formated as elie.field.splitnumber.split so the above example matches all files that match ./tmp/elie.speaker.*.split

```
evaluation.py -t ./train.preprocessed -D ./tmp
     -s ./tmp/elie.speaker.[1-5] -f 'speaker stime etime location'
```

We can also add regular expressions to the splitfile base. The above example matches splitfiles where the base is elie.speaker. and the split number starts with 1, 2, 3, 4 or 5.

After running the above experiment all the log files will be stored in ./tmp. Once the experiment is complete we can use scorer.py to examine the performance. To view the L1 performance we issue the command:

```
scorer.py ./tmp/elie.speaker.*.elie.L1.log
```

To view the L2 performance we would use the following command:

```
scorer.py ./tmp/elie.speaker.*.elie.L2.log
```

# Bibliography


[1] Pascal challenge proposal. *http://www.pascal-network.org/Challenges/EIRD/*.

[2] Pascal challenge website. *http://tyne.shef.ac.uk/Pascal*.

[3] Rehan Akbani, Stephen Kwek, and Nathalie Japkowicz. Applying support vector machines to imbalanced datasets. In *European Conference on Machine Learning*, 2004.

[4] Tim Berners-Lee. Semantic web road map. *http://www.w3.org/DesignIssues/Semantic.html*, 1998.

[5] A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. In *Proc. Workshop Computational Learning Theory*, 1998.

[6] Eric Brill. Some advances in transformation-based part of speech tagging. In *American National Conference on Artifical Intelligence*, 1994.

[7] C. J. C. Burgess. A tutorial on support vector machines for pattern recognition. In *Data mining and knowledge discovery*, volume 2, 1998.

[8] Mary Elaine Califf and Raymond J. Mooney. Relational learning of pattern-match rules for information extraction. In *American National Conference on Artifical Intelligence*, 1999.

[9] N. Chawla, K. Bowyer, K. Hall, and W. Kegelmeyer. Smote: Synthetic minority over-sampling technique. In *Journal of Artificial Intelligence Research*, volume 16, pages 321–357, 2002.

[10] Hao Chen and Susan T. Dumais. Bringing order to the Web: automatically categorizing search results. In *ACM International Conference on Human Factors in Computing Systems*, Den Haag, NL, 2000. ACM Press, New York, US.

[11] Philipp Cimiano, Siegfried Handschuh, and Steffen Staab. Towards the self-annotating web. In *World Wide Web Conference*, 2004.

[12] Fabio Ciravegna. Adaptive information extraction from text by rule induction and generalisation. In *International Joint Conference Artificial Intelligence*, 2001.

[13] William Cohen. Fast effective rule induction. In *International Conference on Machine Learning*, 1995.

[14] William W. Cohen and Yoram Singer. A simple, fast, and effective rule learner. In *American National Conference on Artifical Intelligence*, 1999.

[15] C. Cortes and V. Vapnik. Support vector networks. In *Machine Learning*, volume 20, 1995.

[16] Scott C. Deerwester, Susan T. Dumais, Thomas K. Landauer, George M. Furnas, and Richard A. Harshman. Indexing by latent semantic analysis. *Journal of the American Society of Information Science*, 41(6), 1990.

[17] C.L. Blake D.J. Newman, S. Hettich and C.J. Merz. UCI repository of machine learning databases, 1998.

[18] D. Freitag. Toward general-purpose learning for information extraction. In *35th Annual Meeting of the Association for Computational Linguistics*, 1998.

[19] Dayne Freitag. *Machine Learning for Information Extraction in Informal Domains*. PhD thesis, Carnegie Mellon University, 1998.

[20] Dayne Freitag and Nicholas Kushmerick. Boosted wrapper induction. In *Proc. 17th Nat. Conf. Artificial Intelligence*, 2000.

[21] Dayne Freitag and Andrew McCallum. Information extraction with HMMs and shrinkage. In *AAAI Workshop on Machine Learning for Information Extraction*, 1999.

[22] Alfio Massimiliano Gliozzo, Claudio Giuliano, and Raffaella Rinaldi. Instance pruning by filtering uninformative words: An information extraction case study. In *International Conference on Intelligent Text Processing and Computational Linguistics*, 2005.

[23] R. C. Holte. Very simple classification rules perform well on most commonly used datasets. *Machine Learning*, 11, 1993.

[24] Neil Ireson. An introduction to the pascal challenge on evaluating machine learning for information extraction. In *First Pascal Challenge Workshop*, 2005.

[25] Neil Ireson and Fabio Ciravegna. Pascal challenge: The evaluation of machine learning for information extraction. In *Dagstuhl Workshop on Machine Learning for the Semantic Web*, 2005.

[26] Neil Ireson, Fabio Ciravegna, Marie Elaine Califf, Dayne Freitag, Nicholas Kushmerick, and Alberto Lavelli. Evaluating machine learning for information extraction. In *International Conference on Machine Learning*, 2005.

[27] Thorsten Joachims. A probabilistic analysis of the Rocchio algorithm with TFIDF for text categorization. In *International Conference on Machine Learning*, Nashville, US, 1997.

[28] Thorsten Joachims. Text categorization with svm: Learning with many relevent features. In *European Conference on Machine Learning*, 1998.

[29] John Lafferty, Andrew McCallum, and Fernando Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *International Conference on Machine Learning*, 2001.

[30] A. Lavelli, M.-E. Califf, F. Ciravegna, D. Freitag, C. Giuliano, N. Kushmerick, and L. Romano. IE evaluation: Criticisms and recommendations. In *AAAI Workshop Adaptive Text Extraction and Mining*, 2004.

[31] Alberto Lavelli, Mary Elaine Califf, Fabio Ciravegna, Dayne Freitag, Claudio Giuliano, Nick Kushmerick, and Lorenza Romano. A critical survey of the methodology for IE evaluation. In *International Conference on Language Resources and Evaluation*, 2004.

[32] David D. Lewis. Naive (Bayes) at forty: The independence assumption in information retrieval. In *European Conference on Machine Learning*, Chemnitz, DE, 1998.

[33] Nick Littlestone. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning*, 2(4), 1988.

[34] Andrew McCallum, Kamal Nigam, Jason Rennie, and Kristie Seymore. A machine learning approach to building domain-specific search engines. In *International Joint Conference on Artificial Intelligence*, 1999.

[35] Dunja Mladenić. Turning YAHOO! into an automatic Web page classifier. In *European Conference on Artificial Intelligence*, Brighton, UK, 1998.

[36] I. Muslea, S. Minton, and C. Knoblock. Selective sampling with reduntant views. In *American National Conference on Artifical Intelligence*, 2000.

[37] Fuchun Peng and Andrew McCallum. Accurate information extraction from research papers using conditional random fields. In *Human Language Technology Conference and North American Chapter of the Association for Computational Linguistics*, 2004.

[38] John C. Platt. Fast training of support vector machines using sequential minimal optimization. In *Advances in Kernel Methods - Support Vector Learning*. MIT Press, 1998.

[39] Ross Quinlan. Induction of decision trees. In *Machine Learning*, volume 1, 1986.

[40] Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufman, 1993.

[41] Dan Roth. Learning to resolve natural language ambiguities: A unified approach. In *American National Conference on Artifical Intelligence*, 1998.

[42] Dan Roth and Wen-Tau Yih. Relational learning via propositional algorithms: An information extraction case study. In *International Joint Conference on Artificial Intelligence*, 2001.

[43] Mehran Sahami, Susan Dumais, David Heckerman, and Eric Horvitz. A bayesian approach to filtering junk E-mail. In *Learning for Text Categorization: Papers from the 1998 Workshop*. AAAI Technical Report WS-98-05, 1998.

[44] Fabrizio Sebastiani. Machine learning in automated text categorization. In *ACM Computing Surveys*, 2002.

[45] An De Sitter and Walter Daelemans. Information extraction via double classification. In *Proceedings of the ECML/PKDD 2003 Workshop on Adaptive Text Extraction and Mining*.

[46] Ian H. Witten and Eibe Frank. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann, 2000.