Automated Timeline Extraction via a Semi-supervised Learning Approach

Oulin Yang

A subthesis submitted in partial fulfillment of the degree of
Master of Computing (Honours) at
The Department of Computer Science
Australian National University

October 2011

Typeset in Palatino by TEX and LATEX 2 .

Except where otherwise indicated, this thesis is my own original work. And all experimental data presented in this thesis are real.

Oulin Yang
27 October 2011

To my parents, who offered me full support of my study at ANU
To Grace, her accompany comforts me though there are oceans between us
To Noah, the baby of Scott, to celebrate his birth
And all the best to my grandparents.

Acknowledgements

Above all, I wish to express my sincere thanks to my supervisor Dr.Scott Sanner, who gives me valuable guidances and inspirations in this project also in my academic life. He is a good mentor, a close friend, and a good father. He shares a strong sense of responsibility, wisdom and cares with the people around him. It is my pleasure to be his student.

And a big thank you to the Department of Computer Science and NICTA. They offered me nice working sites and machines.I would also like to extend my appreciations to all the academic staff of ANU and NICTA, my colleagues and friends who have discussed any idea related to this project and offered encouragements to me in the middle of this project.

Accompanied with this thesis, the Java artefact developed by myself is partially functioned by the Document Analysis toolkit owned by NICTA. And the TimeML data parsing functions in this artefact is developed on the basis of TimeML Parsers of the TimeML organisation.

Abstract

This thesis relates to a new developing topic in the Natural Language Processing area of recent years - extracting time and events from news articles. I developed a Java artefact, which is trained and evaluated on tagged news corpus via both supervised and semi-supervised learning methods.

Specifically, this is done first through four baseline algorithms: a 'Dictionary Lookup' classifier, variants of Naive Bayes model, the TARSQI toolkit and the Conditional Random Fields (CRFs) classifier. The experimental results show that, by applying tokenization, part-of-speech tagging, preceding adjacent labels and other features, one variant of Naive Bayes algorithm gives better performance than other variants. Moreover, the CRF classifier outperforms other baselines significantly and beats the 'state-of-the-art' toolkit.

To address the bottleneck brought by the limited size of training data set, this thesis introduces a novel framework of applying co-training algorithm on various combinations of CRFs with distinct feature sets, to automatically generate high quality labelling outputs. The labelled data set produced from co-training's peak performance has been experimented with baseline classifiers. As a result, baselines are performing stronger with the new training data set than with the original data set.

Overall, the major contribution of this thesis is finding an applicable way to use co-training algorithm to process sequential text data, and promote general classification performance over unlabelled data. By this method, scarcity of training data in many circumstances can be eased to some extent.

Contents

Chapter 1

Introduction

1.1 Motivations

Every day news headlines are attracting the attention of the world and touching people's nerves. On the Internet, one can always expect spectacular and rapid surges of the search trends of some popular key words/phrases. Imagine yourself searching for such a key phrase like the 'European debt crisis', 'Libya war','Japan earthquake' or 'Deepwater Horizon oil spill' on any current news search service, typically you will be given thousands or more results (news web pages) of the past few months or earlier. For instance, totally 29,100 news articles are returned by searching 'European debt crisis' on the Google News (tested on Oct 4, 2011). But what if your purpose is to understand the entire story, preferable against a timeline, of what has happened to date as comprehensive as possible in a relative short time? Figure 1.1 illustrates such a timeline which displays some of the major events related to the topic 'Japan Earthquake' happened between 14^{th} Mar and 20^{th} Mar of 2011. An ideal blueprint of this application is capable to extract the most relevant events to the topic from hundreds to thousands of news web pages, so that through the timeline readers can develop a coherent and precise knowledge about the topic in an efficient manner.

With regards to the results returned by a search service, they have been already beautifully and precisely ranked from the most relevant to the least by the latest Information Retrieval techniques. The truth is, however, in most cases readers are not able or reluctant to go through even the very top 1% contents on that long list, however by reading through the first few articles alone is not sufficient to form a holistic picture of the scenario. Furthermore, it requires non-trivial efforts from readers to identify, rearrange and integrate useful information from a detailed news report, let alone from many of them.

Normally people tend to use some encyclopedia-like online services when they encounter unfamiliar topics, such as using Wikipedia - if a specific Wiki page has been created for it by the community. However, a well-structured, accurate and complete Wiki page about a critical event may not be available or reliable in a short time after the event happens. Other alternative solutions could be searching for a well written summary from blogs, asking in forums, or browsing news videos on Youtube. Again,

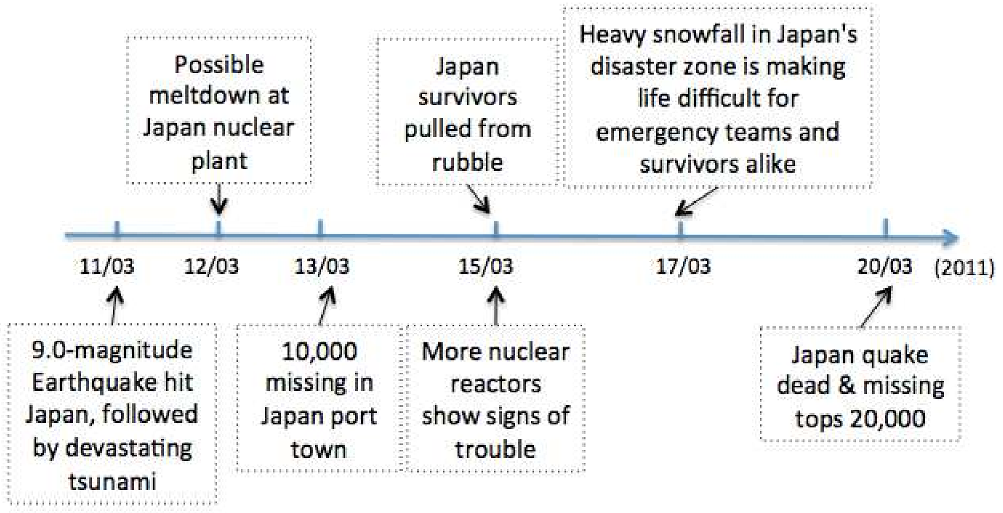

Possible meltdown at Japan nuclear plant
Japan survivors pulled from rubble
Heavy snowfall in Japan's disaster zone is making life difficult for emergency teams and survivors alike
11/03
12/03
13/03
15/03
17/03
20/03
(2011)
9.0-magnitude Earthquake hit Japan, followed by devastating tsunami
10,000 missing in Japan port town
More nuclear reactors show signs of trouble
Japan quake dead & missing tops 20,000

Figure 1.1: An example of timeline displaying of some of major events manually extracted from news reports about the '2011 Japan earthquake' natural disaster

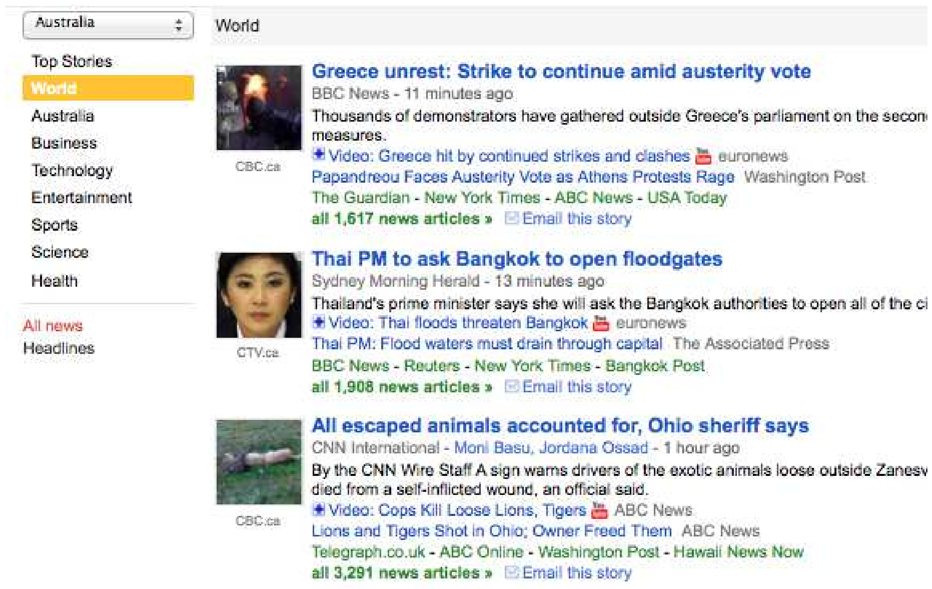
those suggestions are time-consuming and user satisfaction is not guaranteed.

Fortunately, there exists many efforts trying to meet similar market needs. For instance, the Google News provides users with a 'timeline view' function by which articles about a certain topic are grouped together and presented in a chronological order (see Figure 1.2). Still, there are over thousand of results returned of one headline, and readers need to go through many of them before understanding what has exactly happened.

Under this context, this thesis aims to explore an automated framework which extracts time and events from news content, in the hope of later combining those extracted information with their context to build an off-shelf tool, which can quickly analyse the text content of news articles and display a timeline along with major events for readers.

Simply put, the key insights and research contributions of this thesis are:

- Suggestions of how to select good features for Naive Bayes model in this labelling task;
- Comparisons of several baseline algorithms with the state-of-the-art toolkit TARSQI of this task, and the CRF model outperforms TARSQI significantly;
- A successful attempt of applying co-training algorithm to learn and predict unlabelled sequential data to address the limited size of training data, where two independent CRF models achieve around 2% increases in F-scores;



Greece unrest: Strike to continue amid austerity vote
BBC News - 11 minutes ago
Thousands of demonstrators have gathered outside Greece's parliament on the secon
measures.
Video: Greece hit by continued strikes and clashes euronews
Papandreou Faces Austerity Vote as Athens Protests Rage Washington Post
The Guardian - New York Times - ABC News - USA Today
all 1,617 news articles » Email this story

Thai PM to ask Bangkok to open floodgates
Sydney Morning Herald - 13 minutes ago
Thailand's prime minister says she will ask the Bangkok authorities to open all of the ci
Video: Thai floods threaten Bangkok euronews
Thai PM: Flood waters must drain through capital The Associated Press
BBC News - Reuters - New York Times - Bangkok Post
all 1,908 news articles » Email this story

All escaped animals accounted for, Ohio sheriff says
CNN International - Moni Basu, Jordana Ossad - 1 hour ago
By the CNN Wire Staff A sign warns drivers of the exotic animals loose outside Zanesv
died from a self-inflicted wound, an official said.
Video: Cops Kill Loose Lions, Tigers ABC News
Lions and Tigers Shot in Ohio; Owner Freed Them ABC News
Telegraph.co.uk - ABC Online - Washington Post - Hawaii News Now
all 3,291 news articles » Email this story

Figure 1.2: A screenshot of the Google News where news web pages are ordered by recency (on 14th Oct 2011)

- With new data labelled from co-training, there are noticeable improvements in the performances of baseline classifiers than they previously did.

1.2 A Theoretical Framework

The following sections of this thesis detail my efforts of achieving this goal. In the starting part (Section 2) it briefs previous achievements and limitations in similar researches, and then the Natural Language Processing techniques applied. Also it contains an introduction of necessary background knowledge about the machine learning techniques employed in this project, such as Naive Bayes models, Hidden Markov Models and Conditional Random Fields of supervised learning, and Co-training algorithm of semi-supervised learning.

As an important role of this labelling task, in Section 2 there is a detailed explanations about the Time Markup Language - the annotation schema of the corpus adopted as training data. Further, the state-of-the-art annotation tool of this markup language, and the modelling process of this task.

Section 3 explains how time and event are extracted from text content, and feature selections for the machine learning algorithms. Four various baseline algorithms/toolkits are implemented, and the results are carefully compared and discussed.

Section 4 details the implementation of co-training, a semi-supervised learning method, to solve the bottleneck problem with regards to the limited size of training data. I present a co-training framework on CRFs with different models combinations, followed by analysis and discussion of experiment results. To verify if the new labelled data from co-training's peak performance are of values, the new data set is tested on baseline algorithms to compare with previous statistics.

Finally, in Section 5 the thesis concludes with a summary of the positive results produced by the devised co-training framework, and my suggestions for continuing this work in the future.

Chapter 2

Literature Review

With the development of the Internet and Information Technology, people find there are greater potential values and more powerful tools to process usually a huge volume of text of human language and extract useful information from them. Linguistics, statistics, computer science, and techniques from many other fields are combined together to process more complex tasks in a parallel and more intelligent manner. Now machines are not only able to help human to process basic linguistic tasks, but are also capable to learn from past experiences to find out hidden patterns from new data without specific programming.

To identify time and events from a document, some 'linguistic signals' are of great importances. In many cases, tokenization is the very first step to obtain statistical linguistic information, as by that text is parsed into unified units. After tokenization, there are more techniques can be applied to extract further linguistic information, like part-of-speech tagging, context analysis, text mining etc. The importances of those NLP techniques in this project are illustrated in the following sections.

2.1 Natural Language Processing

Natural Language Processing (NLP) is such a combined field of computer science and linguistics concerned with the interaction between machine and human language. Starting from 1950s, many NLP techniques have been developed and successfully applied when dealing with realistic problems, like the application of the WordNet in building user model for websites [Stefani and Strapparava 1999], and the NLP annotating techniques in bio-textmining [Kim et al. 2003].

Usually the first thing in a NLP task is sentence extraction. It is a necessary preprocessing stage for further text analysis as there are often some linguistic constraints regarding to sentence structures. Sentence extraction is not simply the process of taking a paragraph or a document and dividing it into sentences by full stops. It seems a trivial step however sentence extraction may become difficult when determining whether a period is part of an abbreviation, or an actual end-of-sentence.

Tokenization is the process of breaking each sentence into several independent tokens. Tokens consist of words, punctuation, symbols, phrases or other meaningful elements etc. that are helpful for processing the whole sentences or text stream. Usually tokenization is performed at the word level. For example, the sentence 'President Bush today denounced Saddam's "ruinous policies of war".' is tokenised as (each independent token is separated by double forward slashes):

President // Bush // today // denounced // Saddam // 's // " // ruinous // policies // of // war // " // .

Via tokenization, the abbreviation ''s' is separated from preceding word and expanded to an independent token.

Another important technique used in this task is Part-Of-Speech (POS) Tagging. POS tagging is the process of marking each token with corresponding part-of-speech tag (Table 2.1) based on both the token's definition and its context. Currently POS tagging is mostly done with the aid of softwares using algorithms like Hidden Markov Model, dynamic programming, Nearest-Neighbouring, Support Vector Machine, etc.

A number of state-of-the-art NLP softwares and tools are available for various purposes, such as OpenNLP, AlchemyAPI, LinguaStream, Stanford NLP etc. OpenNLP, for instance, hosts a variety of Java-based NLP tools which perform sentence detection, tokenisation, POS-tagging, chunking and parsing, named-entity detection, and co-referencing using the OpenNLP Maxent machine learning package. The NLP techniques applied in this project is primarily based on OpenNLP.

2.2 The Time Markup Language

Many individuals and research groups have made contributions in temporal information extraction. In application, except for Google News (Figure 1.2), the SIMILE Widgets Timeline provides an innovative and interactive web widget for visualizing temporal data (SIMILE Widget Timeline 2006). From the demo on SIMILE Widgets' main page (Figure 2.1), readers can review the history of 'John F. Kennedy assassination' from a fresh perspective: each individual event related to this phrase is summarized into a short sentence, while full details are still accessible by clicking on this one-sentence summary. Important incidents and events are labelled in different colors for emphasis. And user can view the time/dates of their interest against a timeline by dragging the widget horizontally. Overall, this timeline widget allows users to recollect or understand the entire context and the potential relationships between events quickly. As far as I concerned, the SIMILE Widget is almost a developed and feature-rich timeline tool, in terms of software engineering perspective.

Number	Tag	Description
1	CC	Coordinating conjunction
2	CD	Cardinal number
3	DT	Determiner
4	EX	Existential there
5	FW	Foreign word
6	IN	Preposition or subordinating conjunction
7	JJ	Adjective
8	JJR	Adjective, comparative
9	JJS	Adjective, superlative
10	LS	List item marker
11	MD	Modal
12	NN	Noun, singular or mass
13	NNS	Noun, plural
14	NNP	Proper noun, singular
15	NNPS	Proper noun, plural
16	PDT	Predeterminer
17	POS	Possessive ending
18	PRP	Personal pronoun
19	PRP$	Possessive pronoun
20	RB	Adverb
21	RBR	Adverb, comparative
22	RBS	Adverb, superlative
23	RP	Particle
24	SYM	Symbol
25	TO	to
26	UH	Interjection
27	VB	Verb, base form
28	VBD	Verb, past tense
29	VBG	Verb, gerund or present participle
30	VBN	Verb, past participle
31	VBP	Verb, non-3rd person singular present
32	VBZ	Verb, 3rd person singular present
33	WDT	Wh-determiner
34	WP	Wh-pronoun
35	WP$	Possessive wh-pronoun
36	WRB	Wh-adverb

Table 2.1: Alphabetical list of part-of-speech tags used in the Penn Treebank Project

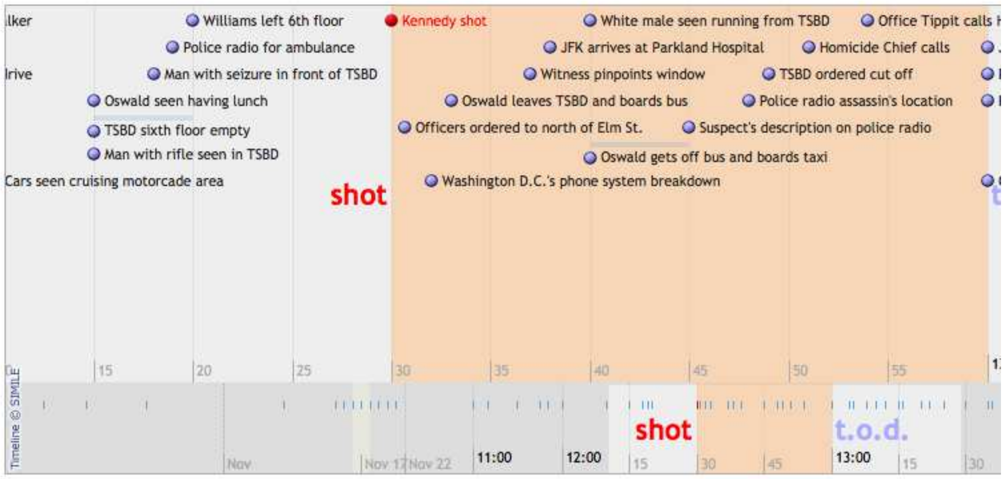

Williams left 6th floor
Kennedy shot
White male seen running from TSBD
Office Tippit calls
Police radio for ambulance
JFK arrives at Parkland Hospital
Homicide Chief calls
Man with seizure in front of TSBD
Witness pinpoints window
TSBD ordered cut off
Oswald seen having lunch
Oswald leaves TSBD and boards bus
Police radio assassin's location
TSBD sixth floor empty
Officers ordered to north of Elm St.
Suspect's description on police radio
Man with rifle seen in TSBD
Oswald gets off bus and boards taxi
Cars seen cruising motorcade area
Washington D.C.'s phone system breakdown
shot
t.o.d.
Timeline © SIMILE
11:00
12:00
13:00

Figure 2.1: A screenshot of the SIMILE Widgets Timeline (on 21th Oct 2011)

Maybe the most attractive accomplishment in temporal information extraction is a robust specification language for events and temporal expressions in natural language called Time Markup Language (in short, TimeML). Based on Allen's relations [Allen 1983], TimeML has been developed in the context of three AQUAINT workshops and projects since 2002 and now is a dominant specification. It is designed to address four problems in event and temporal expression markup: (1) time stamping of events, (2) ordering events with respect to one another, (3) reasoning with contextually underspecified temporal expressions (such as 'last week'), and (4) reasoning about the persistence of event [Pustejovsky et al. 2003].

There are four major annotating structures that are specified in TimeML: *EVENT*, *TIMEX3*, *SIGNAL* and *LINK*. Basically the previous two stands for 'event' and 'time' respectively, and the latter two indicate the relationships among those tags. For details about the definitions, annotation scheme and usage, please refer to the TimeML Annotation Guidelines Version 1.2.1 [Saurí et al. 2006] and [Pustejovsky et al. 2003].

To better understand the schema of TimeML, here is an one-sentence example with *EVENT* and *TIMEX3* annotations excerpted from a document in the gold-standard corpus, where the key words of event and time are started with TimeML tags:

```
"On the other hand, it's"
[EVENT class=OCCURRENCE eid=e1] "turning"
"out to be another very"
[EVENT class=STATE eid=e369] "bad"
"financial"
[TIMEX3 functionInDocument=NONE temporalFunction=false
tid=t83 type=DURATION value=P1W] "week"
```

```
    "for Asia."
```

In TimeML, events are sub-categorised into the following classes: occurrence, perception, reporting, aspectual, state, intensional state, intensional action, and modal [Pustejovsky et al. 2003]. And time related key words are classified by their functions and types. Further discussion about event taxonomy will be out of scope of this thesis.

In this project, a gold standard corpus called TimeBank Corpus which will be elaborated in later chapter is adopted as the only source of training/testing data. And this corpus is under the TimeML annotation schema 1.2.1.

2.3 Overview of Time and Event Extraction

2.3.1 Task

Now it is appropriate to introduce the task of extracting time and events in a more statistical or mathematical manner. In this task the finite target set V contains two TimeML tags *TIMEX3* and *EVENT*, and one created tag *NONE* which represents the predicted tag is neither *EVENT* nor *TIMEX3*. So the objective is to calculate the maximum classification probability of each token in a text content. Generally speaking, a supervised labelling task could be concluded as following procedures:

(1) Perform sentence extraction, tokenisation, POS tagging for each document, and correctly map TimeML tags labelled on words or phrases to tokens.

(2) Train a classifier with the outputs from step 1 (this is also called features construction step as features counting are performed);

(3) Evalute performance of the trained classifier on testing data via K-fold cross validation.

To illustrate the output of labelling after tokenization, here is a toy example that shows a sequential view of TimeML labelling on tokens, with letters E, T and N representing *EVENT*, *TIMEX3* and *NONE* tags respectively. Note this example does not have to be exactly correct with TimeML 1.2.1 annotation guide but just for illustration purpose.

(tokens)	Two	days	ago	we	participated	in	that	ceremony	.
(tags)	*T*	*T*	*T*	*N*	*E*	*N*	*N*	*N*	*N*

For most supervised learning methods, one key point is selecting good features. Making the most 'informative' features for prediction will usually make greater contribution for building a predictor, particularly if the variables are redundant [Guyon and Elisseeff 2003]. Below I will illustrate the process of feature selection in this project, and further explanation of the labelling task.

2.3.2 Features

Feature selection in machine learning is usually a non-trivial stage. Good features sometimes are valued as the soul of a machine learning algorithm. In other words, with a poor feature selection, complicated algorithms such as Hidden Markov Model or Support Vector Machine [Bishop 2006] may give a unacceptable performance. Step (3) described in the prediction task in section 2.3.1 can now be interpreted as 'to forecast the TimeML tag of each token from its surrounding context - where useful features can be extracted too'. Following I will discuss in-depth what each of the features is and why they may be important in prediction.

The first feature I include in classification is the *POS tags*. As mentioned before it has become a necessity in many NLP tasks. As to tokens, POS tags can provide complimentary linguistic information in classification. And according to my observations, most of *EVENT* tags are labelled on verbs. So Part-Of-Speech tagging could be very helpful for TimeML prediction if applied to each token.

Next, for both human and computer, it is always important to know the context when try understanding a specified word in a group of text. That is, when predicting the TimeML tag of a token on position i, it will be useful if its surrounding context (of positions $i \pm k$, where $k \in \{1, 2, 3, ...\}$) is known at the same time. Here I define the concept of the *Window Size* (n-grams), which means when determining the TimeML tag of token on position i, only the content within the 'window' can be seen to the algorithm. And then the algorithm scans each token while moving the 'window' forward until the end of text.

The importance of introducing a sliding window is obvious. If the window size equals to zero, that means only the token in current position is visible to the algorithm, and that leads to fewer features are available for the classification task as well. In fact, the result will tend to be biased in this situation. Let's consider the following example:

"A **record** 5.2 million iPhones were activated in the last quarter by AT&T"

and

"However, we **record** the related stock compensation expenses on an accelerated amortisation basis for IFRS reporting. "

If a classifier has already 'seen' the first sentence before, in which the word *record* is a noun and labelled as '*NONE*', then when predicting the same word in the second example where *record* is a verb instead should be tagged with '*EVENT*', the classifier is very likely to make a misjudgement if lacking support from context. We normally won't make this mistake as when looking at previous sentences, as we know for sure that the *record* after the article *a* in the first sentence cannot be a verb, similarly the one after the word *we* in second sentence cannot be a noun too.

Another potential valuable feature I observed from TimeML tagging scheme is the tag itself, i.e. the tags assigned to previous tokens. I name this feature as **preceding adjacent TimeML tags**. Except for part-of-speech tags, this feature could be useful to predict the TimeML tag of the target token. An instance mentioned previously has pointed out that some phrases like:

"one and a half hour ago"

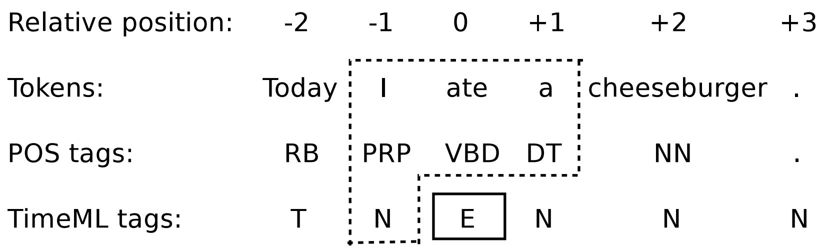
or

"the day before the Final game"

shall be labelled with one *TIMEX3* tag together. However, in practice the classifier will probably treat each of those tokens separately with *NONE* tags, because those single tokens are not as typical as some other *TIMEX3* words like *today* or *Friday*. In this circumstance, making correct prediction of such a group of tokens could become extremely hard if without any knowledge about previous TimeML tags. But by introducing the 'preceding adjacent TimeML tags' feature, this problem may be largely eliminated. In the first example sentence, if tokens like 'one', 'and', 'a' and 'half' are sequentially labelled with the *TIMEX3* tag, which is an extreme unusual case as in 95% of chance they are labelled with *NONE*. So from this perspective, I would argue introducing this feature will be helpful.

Nevertheless, we should notice that the improvement of classifier performance brought by 'TimeML preceding adjacent TimeML tags' feature may not be as substantial as having POS tagging or using the optimal window size, as this feature only works in those special occasions as described above which account for a very small portion in the dataset.

To better illustrate this TimeML tag prediction task, Figure 2.2 shows a good example of feature selection from a short sentence. In this example, the size of the sliding window is 1, and features applied to prediction task are POS tags and preceding adjacent TimeML tags. The target tag to be predicted is highlighted within a box of solid line, and the area marked by dash line is all the information used to classify target. At position 0, the current token is 'ate', and it is a verb. I also know its previous token, at position -1 is 'I' whose part-of-speech tag is PRP, and it has been labeled with tag *NONE*. Moreover, the next token is 'a', with part-of-speech tag 'DT'. Note the TimeML tags at positions $+1$ to $+3$ though listed here for example but should be unknown for predictor in real task.



Relative position: -2 -1 0 +1 +2 +3
Tokens: Today I ate a cheeseburger .
POS tags: RB PRP VBD DT NN .
TimeML tags: T N E N N N

Figure 2.2: A simple example of TimeML tag prediction with three features

2.3.3 Methodology

I employed four statistical measures to evaluate the algorithm performance. They are: **precision**, **recall**, **accuracy**, **F$_1$ score** (I will use 'F-score' in all following occurrences) with their **standard errors**. Those measures are calculated by true positive (tp), false positive (fp), true negative (tn) and false negative (fn) of a confusion matrix.

$$precision = \frac{tp}{tp + fp} \tag{2.1}$$

$$recall = \frac{tp}{tp + fn} \tag{2.2}$$

$$accuracy = \frac{tp + tn}{tp + tn + fp + fn} \tag{2.3}$$

$$F_1 score = 2 \times \frac{precision \times recall}{precision + recall} \tag{2.4}$$

As defined in binary classification task, **true positive** means the correct prediction of tokens which should be tagged with *EVENT* or *TIMEX3*; **false positive** means tokens which are tagged with *EVENT* or *TIMEX3* but actually should not be. Similarly, **true negative** refers to those tokens are labelled with *NONE* correctly, and **false negative** are tokens which should be tagged with *NONE* but not.

2.4 TARSQI Toolkit Classifier

TARSQI (Temporal Awareness and Reasoning Systems for Question Interpretation) Toolkit (TTK) is one of the most influential toolkits published by TimeML organisation in November 2007. TTK identifies temporal expressions and events in natural language texts, and parses the document to order and anchor events to temporal expressions. It integrates extraction of events and time expressions with creation of temporal links, using a set of mostly independent modules, while ensuring consistency with a constraint propagation component. The TARSQI Toolkit (TTK) provides a box of functions for temporal needs. The glue that keeps all the modules together is the TimeML language.

This toolkit has been applied in a number of research works like [Ling and Weld 2010]. Within this tool package, *Evita* ('Events In Text Analyzer') is quite notable, as an event recognition system that has achieved a performance ratio of 80.12% F–score in the identification and tagging of event expressions work [Saurí et al. 2005]. For more information about TARSQI, [Verhagen et al. 2005] and [Verhagen and Pustejovsky 2008] elaborate the functions of each component of this toolkit.

As this labelling task is based on the same annotation schema, the TARSQI Toolkit can be considered as the state-of-the-art technique that provides me reliable benchmarks if using the same data to test the algorithm of TTK.

2.5 Supervised Extraction Methods

Supervised classification, or inductive learning, refers to a machine learning task of training on supervised data - it consists of input objects and desired outputs. In a supervised classification task, both training and testing data are labelled with predefined classes. In other words, the algorithm learns a model using the training data, and exams this model with unseen data (testing data) to assess the accuracy of the model by comparing the predicted results with ground truth.

The following sub-sections first introduce the data used through the entire experiment. Each section illustrates the principle of a model and sufficient introduction about how they are implemented in this task. At the end of this part, there is a short discussion about an important problem with current dataset when I try to expand experiments to more general cases.

2.5.1 Dictionary Look-up Classifier

The most straight-forward approach to address a labelling task could be using a dictionary look-up method. This method has nothing to do with learning but very much a means of rote memorisation. Through training samples, the classifier finds out all of the tokens which are labelled with desired tags, and put them in searching efficient data structures (like Hash Maps) by category. The occurrence of each token is incremented if it has appeared more than once during training. So far three 'dictionaries' against TimeML tags *EVENT*, *TIMEX3* and *NONE* have been built up.

If a token has been labelled with different tags in the training samples, this classifier will compare the occurrences of this token with each tag and assign the tag with the highest frequency. If a token has been labelled with different tags but with same frequencies, or even it cannot be found in any 'dictionary record' at all, then the classifier will randomly pick up a tag and assign it to this token.

The objective of introducing this very basic classifier is to provide a baseline to all 'learning' algorithms. Though someone may argue that the performance of a dictionary look-up classifier could be improved significantly if given sufficient large or nearly infinite size of training data theoretically, but it will still fail to label the words that it has never 'seen' before in a reliable manner. The truth is, human's linguistic sources have evolved so quickly over the last decades that one cannot heavily rely on feeding more training data to any model. And as I will discuss later, in many cases extra labelled data set is unavailable or very expensive.

2.5.2 Naive Bayes Classifier

Naive Bayes classifier is a highly practical machine learning model in many domains though the principle behind it looks simple. In practice, it has been successfully applied to natural language processing tasks quite early and frequently e.g. [Androutsopoulos et al. 2000], [Escudero et al. 2000] and [Dumais et al. 1998]. Despite its simplicity, the performance of a Naive Bayes model has been shown to be comparable to that of neural network and decision tree learning in those tasks as mentioned in [Mitchell and Carbonell 1986] and [Manning et al. 2008]. Different from traditional Bayesian approach, the Naive Bayes classifier is based on the simplifying assumption that the values of attributes are conditionally independent given the target values. If I name the most probable target value as v_{MAP} which belongs to some finite set V, and $V = \{EVENT, TIMEX3, NONE\}$, where the third one means the TimeML tag of current token is not any of previous two. And attribute values are $< a_1, a_2, ..., a_n >$, where $a_i \in R$. For example, attributes could be boolean values like whether POS tags is included in prediction, or integer values like the size of sliding window, or real values like the value of smoothing constants. So with previous assumption the format of the Naive Bayes classifier is:

$$v_{NB} = \arg\max_{v_j \in V} P(v_j) \prod_i P(a_i|v_j) \tag{2.5}$$

During the learning process, the prior probability $P(v_j)$ is calculated by counting the frequencies of v_j in the corpus. The likelihood $P(a_i|v_j)$ is learned by computing the division of the joint class label and the feature occurrence. In formula forms,

$$P(v_j) = \frac{Freq(v_j)}{\sum\limits_i Freq(v_i)} \tag{2.6}$$

$$P(a_i|v_j) = \frac{Freq(a_i \wedge v_j)}{Freq(v_j)} \tag{2.7}$$

In evaluating step, the value of v_{NB} will be decided by empirically estimation of $P(v_j)$ and $P(a_i|v_j)$. And my goal is to find the maximum v_{NB} for each token.

For example, to calculate the posterior probability of a verb 'announced' classified as *EVENT*, i.e. P(*EVENT* | announced), firstly the prior probability of TimeML tag *EVENT* is the proportion of the number of tokens with *EVENT* tags out of the number of tokens in the entire corpus. The values of prior probabilities are fixed to a particular dataset before training, i.e. 0.13 for $P(EVENT)$ in this dataset.

Next, I need to calculate the likelihood $P(announced|E)$. Let's say in the training data I have found 31 occurrences where the word 'announced' is labelled with *EVENT* tag, and the total number of tokens with this tag is 7935. In addition, to avoid dividing by zero problems I add a smoothing constant to each probability in the division. Therefore,

$$P(E|announced) = P(E)P(announced|E) = 0.13 * \frac{31 + 0.005}{7935 + 0.005} = 0.0508\%$$

The value of smoothing constant is critical to the overall performance of Naive Bayes classifier but it is very easy to ignore. In experiments I find various values of smoothing constant can lead to up to 10% difference in F-scores. Conventionally, this value sits between 0.001 and 3, but the exact value depends on different sectors and experiences. Here 0.005 is adopted for the smoothing constant, because it allows the Naive Bayes classifier performing best in this prediction task.

Similarly, I calculate the conditional probabilities of another two tags $P(T|announced)$ and $P(N|announced)$. After that, I pick up the conditional probability with the highest value and assign that TimeML tag as the output of the token.

Currently in many machine learning studies of supervised learning, the SVM is becoming a more favourable baseline algorithm than Naive Bayes like [Yanagawa et al. 2007] and [Aytar et al. 2007] because of its more stable performance. Due to time constraints, I am not going to implement and compare all these supervised methods but jump to sequential models in next section.

2.5.3 Conditional Random Field Classifier

For modelling sequential data, Hidden Markov Models (HMMs) [Rabiner and Juang 1986] is a powerful probabilistic tool and have been successfully applied to many language-based tasks, such as speech recognition [Rabiner 1989], information extrac-

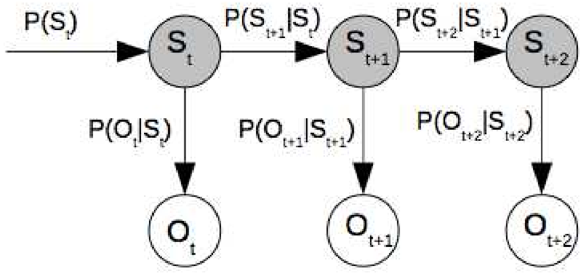


Figure 2.3: The example of HMM graphical model in this task, where the shaded nodes represent hidden states

tion [Freitag and McCallum 1999] and text segmentation [McCallum et al. 2000]. Many sequential data have the Markov property, i.e. the conditional probability distribution of current state of the process depends only upon the most recent state, but not on the sequence of earlier states. The Markov property is one of important characteristics of human language.In this task the prediction of the TimeML tag of a token is highly related to the context close to it but this dependence becomes very week if the distance between the target token and another token is very long. Hidden Markov Model considers joint labeling which means it would automatically handle aspects like 'preceding adjacent TimeML labels'. As HMMs are highly related to CRFs, it is useful to form a concept about Hidden Markov Models for this labelling task to better understand the nature of the Conditional Random Fields applied in this task.

In a HMM model, the observations O_i are words (tokens) that I can directly observe from news articles, and $O_i \in$ Vocabulary. The 'hidden' states S_i are the TimeML tags that I want to model, i.e. $S_i \in \{EVENT, TIMEX, NONE\}$. Figure 2.3 illustrates such a model of three words.

According to the Markov property, the joint probability of this HMM model $P(S_t, S_{t+1}, S_{t+2}, O_t, O_{t+1}, O_{t+2})$ is then calculated by multiplications of the prior, transitional probabilities and emission probabilities:

$$P(joint) = P(S_t)P(O_t|S_t)P(S_{t+1}|S_t)P(O_{t+1}|S_{t+1})P(S_{t+2}|S_{t+1})P(S_{t+2}|O_{t+2}) \quad (2.8)$$

Table 2.2 shows a truth table of the joint probability. The sum of probabilities of all 27 possible label assignments is normalised to 1. Apparently computing such a table will be very expensive: assume the size of label is $|L|$ and the size of vocabulary is $|V|$, then the size of such a table is $|L|^3 \cdot |V|^3 = O(n^6)$.

Fortunately computing the right hand side of the joint probability in formula 2.8 will be much easier than calculating the entire truth table. And the complexity is $|T| + 2|T|^2 + 3|T||V| = O(n^2)$.

S_t	S_{t+1}	S_{t+2}	O_t	O_{t+1}	O_{t+2}	Probability
N	N	N	the	dog	ate	0.315
N	N	E	the	dog	ate	0.569
N	E	E	the	dog	ate	0.064
...	...	...	...	...	...	...
T	T	T	the	dog	ate	0.008

Table 2.2: The truth table of the joint probability of HMM example of total length of 27

In general, a uniform expression of HMM model is

$$P(S_1, S_2, ..., S_n, O_1, O_2, ..., O_n) = \prod_{t=1}^{N} P(S_t|S_{t-1})P(O_t|S_t) \tag{2.9}$$

where N is the total number of observations. And our goal is to find out the maximum likelihood $P(S_1, S_2, ..., S_n, O_1, O_2, ..., O_n)$. Viterbi algorithm is one of the most frequent used algorithms to find out the most likely sequence of hidden states quickly.

There are two characteristics of sequential data: statistical dependencies exist between the targets (I wish to model), and second, each target usually has a rich set of features that can contribute in classification. Traditional graphical models try to model the joint probability distribution $P(S, O)$ where S is the hidden state that I wish to predict, and O represents the observed knowledge about the targets. However, modelling the joint distribution can be difficult when using rich features, as it requires modelling the distribution $P(O)$, which can include complex dependencies.

A solution to this problem is to directly model the conditional probability $P(S|O)$, which is sufficient for classification tasks. Conditional Random Fields [Lafferty et al. 2001] is such a model. Instead of maximise the joint probability, the CRFs learn to **maximise the conditional likelihood**:

$$P(S_1, S_2, ..., S_n|O_1, O_2, ..., O_n). \tag{2.10}$$

The conditional random field is just a conditional distribution $P(S|O)$ with an associated graphical structure [Sutton and McCallum 2006].

Linear-chain CRFs are similar to HMMs as sequence models, which model dependencies only between neighbour labels based on the assumption of Markov process property. But sometimes it is important to model long-range dependencies between targets. A skip-chain CRF is a conditional model of such flexibility, but also not as compute intensive as n-gram models if n is large. It's worthy mentioning that the skip-chain CRFs are the state-of-the-art solution to current information extraction tasks, and there is a growing interest in this model, like [Galley 2006], [Sutton et al. 2004], and [Kim et al. 2010].

In this labelling task the CRF models are implemented by the toolkit CRF++ [Kudo 2007]. CRF++ is an open source implementation of linear-chain CRFs for segmenting/labelling sequential data. It is designed for generic purpose and has been widely applied to a variety of NLP tasks, such as Named Entity Recognition, Information Extraction and Text Chunking.

2.6 Semi-supervised Extraction Methods

Semi-supervised learning is a technique which combines both labelled and unlabelled data for training, which especially suits our case where only a small amount of labelled data is available but with a large amount of unlabelled data.

A typical semi-supervised learning technique is Co-training, in which two or possibly more learners are trained on a set of examples, but with each learner using a different set of features for each example. [Blum and Mitchell 1998] proposed such a framework for classifying web course pages that builds two classifiers: one learns from the words appear on the web page, and another learns from the content of hyperlinks which pointing to that page. Ideally, the more independent the feature sets of each learner are from the other's, the more likely that each learner will benefit more from the other [Nigam and Ghani 2000] demonstrates that algorithms explicitly leverage a natural independent split of the features outperform algorithms that do not in co-training. Several other research efforts have shown successful applications or implementations of co-training framework in processing text, but primarily with non-sequential data, like [Sarkar 2001] presents empirical results which indicate training a statistical parser on the combined labelled data and unlabelled data strongly outperforms training only on the labelled data; and [Xuan-Hieu 2005] presents a co-training approach for CRFs and this approach achieved significant results on noun phrase chunking.

In short, co-training is an algorithm under which each classifier 'learns' how to do the labelling task better from the prediction results of the other classifier. This process is repeated until there is no unlabelled data left or some stopping criterion are satisfied.

For more information about the application and results of semi-supervised learning approach in this task, please refer to Section 4 where co-training is discussed more formally and in-depth.

2.7 Summary

This section covers the background knowledge about three main parts of the project: Natural Language Processing techniques applied, four baseline algorithms in supervised learning, and a short introduction to the semi-supervised learning approach. Experimental results of supervised learning and unsupervised learning algorithms will be analysed in the following Section 3 and Section 4 respectively.

Chapter 3

Experimental Results of Supervised Extraction Methods

3.1 Dataset

The labelling task is based on the TimeBank Corpus (version 1.2) is a collection of news articles in TimeML format (a variant of XML format). It is a gold-standard human-annotated corpus marked up for indicating *event*, *times*, and *temporal relations* [Pustejovsky et al. 2003]. TimeBank Corpus 1.2 contains 183 documents annotated under the TimeML 1.2.1 specification. And the lengths of each document are varied significantly.

A typical TimeML file has three parts. The first heading part contains information like titles, date reported, authors and press names, the TimeML file structure information (often has many meaningless characters) etc. The second part is the text body part with corresponding TimeML tags. The last part of the document is normally occupied by TimeML tags which indicate temporal or logical relationships between tokens, e.g. the tag *TLINK* and *SLINK*.

Some data pre-processing work have been done on the original TimeML data. For example, the structure information in the heading part is pruned carefully. The reason is first this part (except for the report dates which are explicitly labelled with *TIMEX3*) makes little contribution in training a good model however introduces noises. Another reason is the design of TARSQI toolkit (which will discuss shortly) does not take the heading part of TimeML document into consideration but other baselines do. So this modification to original TimeML files can make sure all classifiers are given with the exactly same training data.

Overall, this version of TimeBank Corpus contains over 61,000 non-punctuation tokens, where 7935 tokens (around 13%) are with *EVENT* tags, and 1414 tokens (around 2.3%) are labelled with *TIMEX3* tags. The dataset is randomly split into two parts as a 90:10 ratio for training and testing purposes respectively.

For more details about TimeML and its annotation schema, please refer to Section

2.2.

3.2 Results and Analysis

All experimental results regarding to four baseline classifiers presented in this thesis are averaged by 10-fold cross validation. The advantage of using a cross validation technique in supervised learning tasks is that it can largely avoid over-fitting on training data, i.e. it generalises as well as possible to new data rather than only succeeding on data that have already been seen. Besides, error bars are calculated and marked in all graphs to present statistical information in a more precise manner.

Firstly, let's have a look at the results of the Dictionary Lookup classifier at the first row of table 3.5. Though a high accuracy of 90.32% seems to be useful, actually it is just the consequence of the truth that about 85% of tokens in training data are labelled with TimeML tags neither *EVENT* nor *TIMEX3*. As a result, the number of tokens that are labelled with tag *NONE* (true negative) is much higher than other three parameters (true positive, false negative, and false positive) and that leads to high accuracy values for all baselines. Another general observation from table 3.5 is, there usually exists a trade-off between precision and recall. Thus, the measurement F-score which combines information of precision and recall is becoming more important and informative in this task, and it comes to be the first to be investigated in all results. As expected, almost all measurements of the Dictionary Lookup classifier are lower than others, which demonstrates the innate defect of a rote memorisation method.

Secondly, a series of experiments for the Naive Bayes classifier with various window sizes have been presented (table 3.1 to 3.4). As previously illustrated in section 3.1, a zero value of window size means only the features of current position are considered in prediction, and a window size 1 includes more features at the previous one and the next one neighbouring spots. Similarly, the wider the window size is, the more context information is contained in prediction. However, it is not necessary applying a very wide window size while one may think that will contribute more. On the contrary, it may introduce too many useless patterns and noises and the program may become unnecessary computing intensive.

Window Size	0	1	2	3	4	5
token	**83.37**	58.93	58.26	57.06	55.60	53.97
token+POS	52.06	62.21	62.34	61.57	59.53	58.94
token+POS+prevTML	–	63.96	64.48	63.06	61.27	60.41

Table 3.1: Precision (%) of variants of Naive Bayes classifier of different window sizes. In the first column, the 'token' means only tokens are considered; and 'token+POS' means both tokens and their POS tags are considered; and 'token+POS+prevTML' adds preceding adjacent TimeML tags.

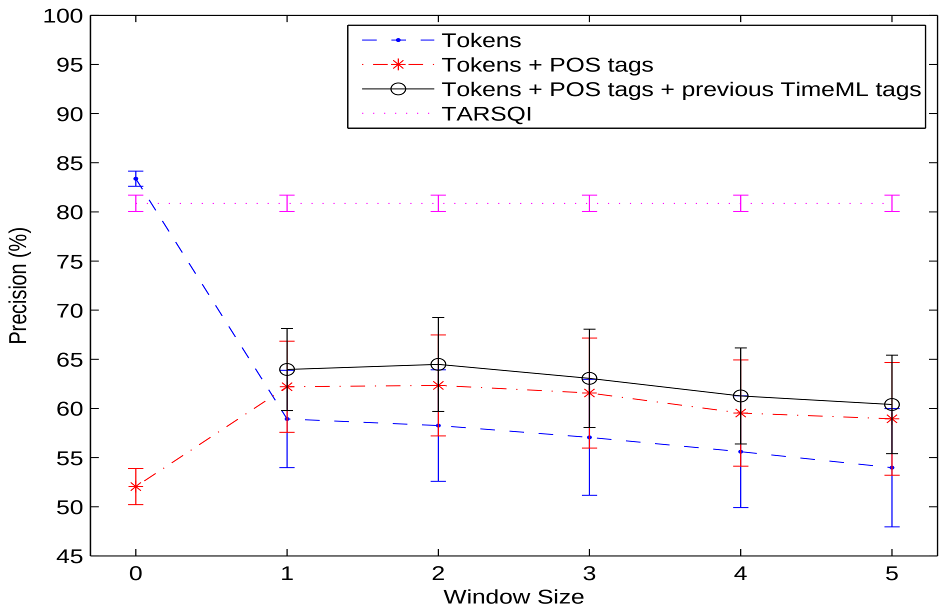


Figure 3.1: The precision values of Naive Bayes classifier

Window Size	0	1	2	3	4	5
token	62.23	81.04	77.11	72.92	68.75	65.41
token+POS	81.03	85.02	82.65	79.21	75.30	72.55
token+POS+prevTML	–	**85.66**	83.04	79.65	76.05	73.41

Table 3.2: Recall (%) of variants of Naive Bayes classifier of different window sizes

Window Size	0	1	2	3	4	5
token	91.62	87.34	86.84	86.18	85.49	84.72
token+POS	84.60	88.89	88.68	88.19	87.31	86.87
token+POS+prevTML	–	**89.59**	89.46	88.77	87.93	87.48

Table 3.3: Accuracy (%) of variants of Naive Bayes classifier of different window sizes

From table 3.4 I notice that when the size of sliding window is 1, and the Part Of Speech tags and the TimeML tags are included, the Naive Bayes classifier makes the strongest performance in this labelling task and achieves a 0.7250 in F-score. This value is higher than 0.7040 at window size 0 (in fact that is very much a Dictionary

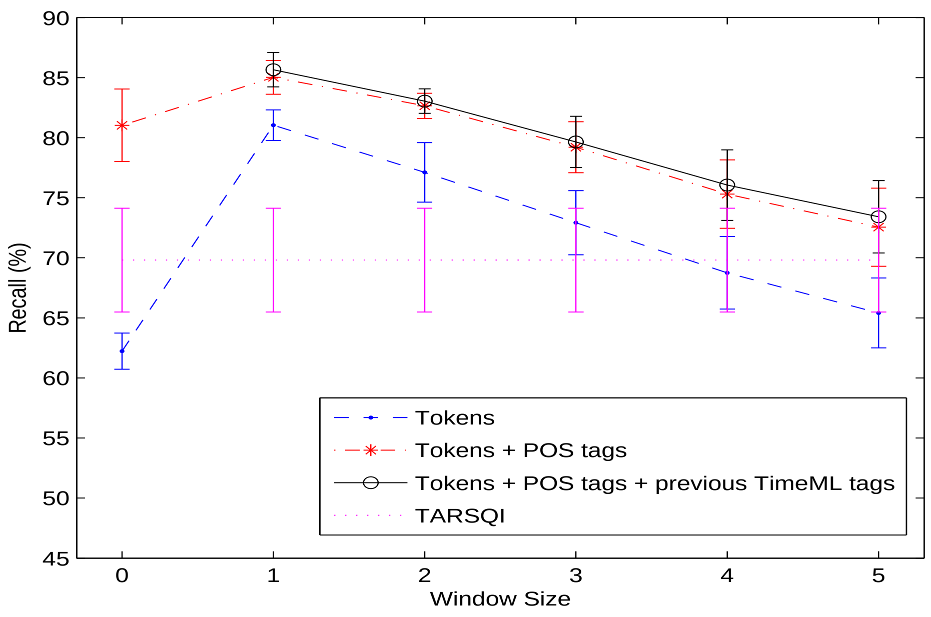


Figure 3.2: The recall values of Naive Bayes classifier

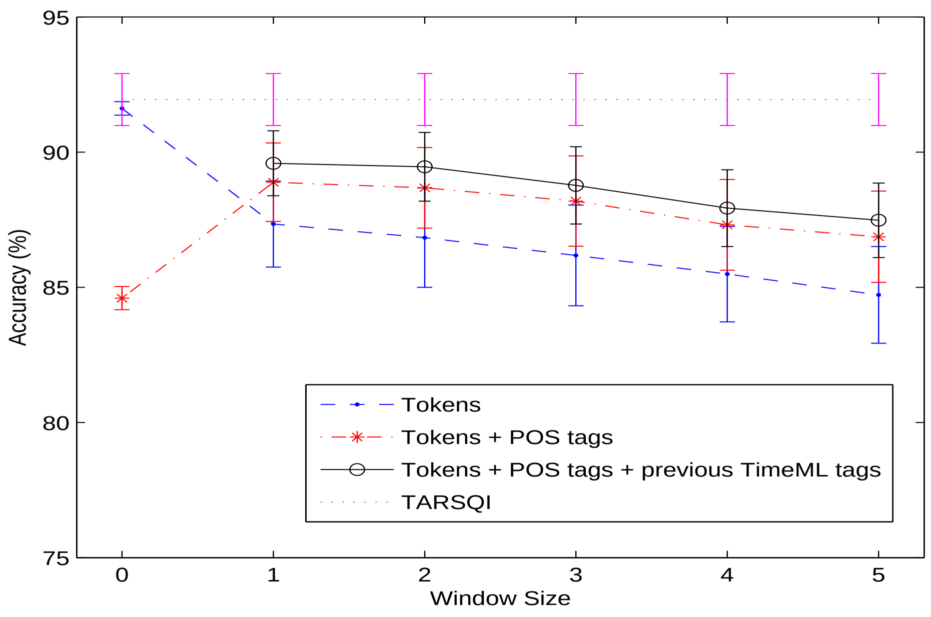


Figure 3.3: The accuracy values of Naive Bayes classifier

Lookup way as no context or other features are considered), which justifies that the classifier is benefit from learning surrounding information. Also, when the window size is increased to 2 with the same features, the F-score slight drops but maintains to

Window Size	0	1	2	3	4	5
token	0.7040	0.6727	0.6536	0.6298	0.6024	0.5786
token+POS	0.6253	0.7102	0.7018	0.6838	0.6552	0.6398
token+POS+prevTML	–	**0.7250**	0.7179	0.6959	0.6692	0.6527

Table 3.4: F-scores of variants of Naive Bayes classifier of different window sizes

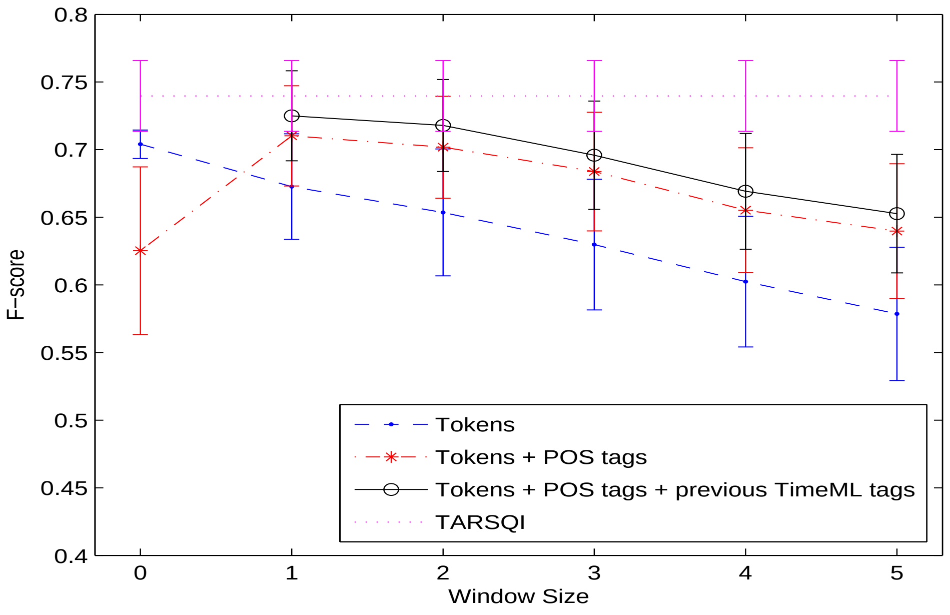


Figure 3.4: The F-score values of Naive Bayes classifier

be the second best score in the table. That means setting the size of sliding window to 2 is also acceptable but may not work as well as window size 1. Later when the size of sliding window becomes larger and larger, all four measurements have shown an obvious downward trend as expected.

The way to examine TARSQI is assigning the same testing data to it to see how its inside 'built-up' model works with the testing data. The results turn out that TARSQI does better than both Dictionary Lookup and Naive Bayes classifiers in terms of F-scores, however the advantages over Naive Bayes are subtle and this is not as good as I expected for a state-of-the-art toolkit.

The results of CRF bring me more excitement among all four baselines. In CRF++ the feature setting is presented in a 'template' file to training CRF model, so how to generate a template will make a lot of differences in outputs. According previous experiences from the Naive Bayes model, it appears that in this task choosing window size 1, combining part-of-speech tags and preceding adjacent TimeML tags could be optimal feature selection in current configuration. So CRF classifier uses the same feature settings as the best variant of Naive Bayes model, and the Bigram template is adopted.

The training units to CRFs can either be documents or sentences, and that makes important differences in co-training in Chapter 4. So here training results on both documents and sentences are given in table 3.5 .

Classifiers	Precision (%)	Recall (%)	Accuracy (%)	F-score
Dictionary L.	79.63 (± 0.96)	56.30 (± 0.87)	90.32 (± 0.47)	0.6517 (± 0.0065)
Naive Bayes*	63.96 (± 4.17)	**85.66** (± 1.43)	89.59 (± 1.20)	0.7250 (± 0.0333)
TARSQI	80.87 (± 0.83)	69.81(± 4.32)	91.95(± 0.96)	0.7397(± 0.0026)
CRF (doc)	**90.99**(± 0.78)	80.06(± 4.03)	95.34(± 0.55)	**0.8467**(± 0.0274)
CRF (sent)	90.91(± 0.79)	80.13(± 4.11)	**95.35**(± 0.56)	**0.8467**(± 0.0278)

Table 3.5: Average performance of four baseline classifiers

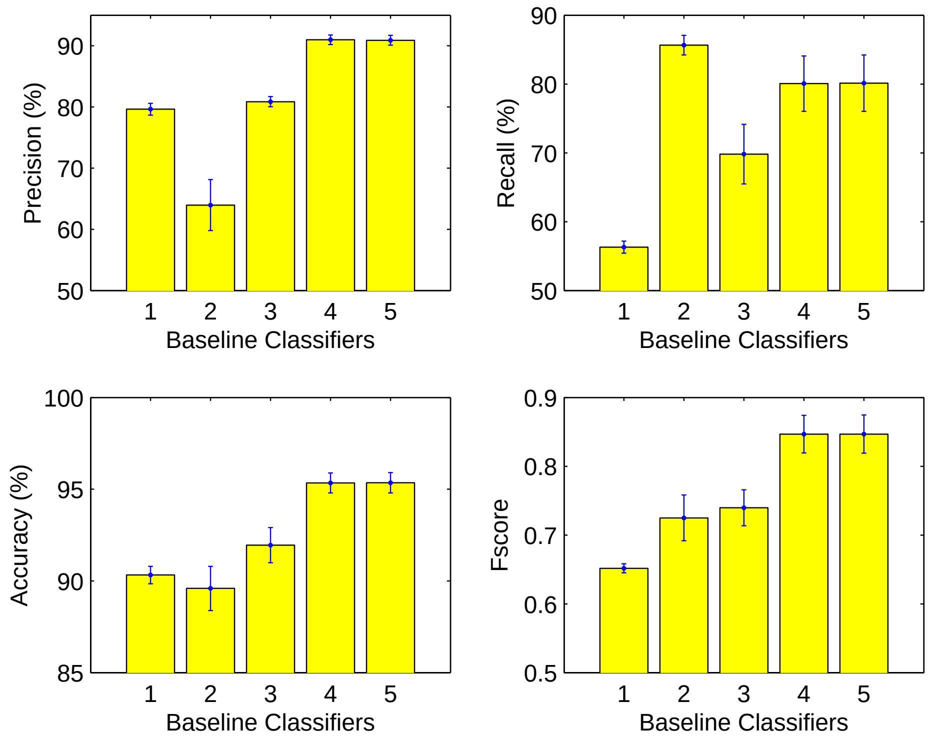


Figure 3.5: The bar charts of four baseline algorithms on different measurements. The numbers on the X-axis, from left to right, represents: 1. Dictionary Lookup Classifier; 2. Naive Bayes Classifier of the optimal feature set; 3. TARSQI; 4. CRF Classifier on documents; 5. CRF Classifier on sentences. Except for recall, CRF models outperform other baselines.

The four bar charts in Figure 3.5 illustrate the content of table 3.5 from a more sensible view. The reason why CRF models perform better is it takes more context information into consideration and filters out irrelevant information when making predictions, rather than based on current position only and equally treat all its neighbouring text as Naive Bayes model does.

3.3 Summary

Overall, CRF classifier significantly outperforms the other three classifiers almost in all measurements except for recall, where the Naive Bayes maintains the highest 85.66% whereas suffers from the lowest precision values. In general, CRF is a very robust classifier in this labelling task either training on documents or sentences, as both precision and recall are above 80%, and that leads to a 0.8467 F-score. This score is almost 11% higher than TARSQI's.

Though now I have a better model which beats the state-of-the-art toolkit, any attempt to further improve current results will be limited by from the insufficient size of training data (only 181 documents are available in TimeBank Corpus). In next section, I will introduce a way to solve this problem via semi-supervised learning.

Chapter 4

Experimental Results of Semi-supervised Extraction Methods

From previous results of supervised learning algorithms, though the Conditional Random Field classifier achieves a very impressive performance of 0.8467 F-score, however, the limited size of training data greatly hindered further development of this extractor. In other words, if applied to process larger amount of sequential data, e.g. predict time and events on thousands of documents per day where news articles are updated on a daily basis, it becomes almost impossible for this extractor to improve or even maintain its performance with current achievements, as more and more new words and expressions are emerging from the Internet but unfortunately the application has already lost its ability to 'learn'. Even though features like part-of-speech tagging and context will always contribute in classifying new words, its performance will gradually drop and finally the extractor could become obsolete.

Intuitively, one solution is to increase the size of training data and try to keep them up-to-date. This looks like a straight-forward way to address current issue but actually it is a very expensive process or sometimes even inapplicable. In this task, for instance, there is no further version of the TimeBank Corpus issued many years after version 1.2, and it seems that the TimeML group has no intention to do so. Also, we can imagine that having human annotators gathering together to perform high standard coups annotation task would cost considerably from both labour and financial perspectives. More importantly, this is not the fundamental solution to a sequential data labelling task, as it does not provide an effective solution to process unknown data.

Unsupervised learning is a machine learning technique which aims to find hidden patterns in unlabelled data. This technique is not limited to the size of dataset but there is no way to make use of the labelled corpus at all. Reinforcement learning is a similar technique with an error/reward signal to evaluate a potential action by an actuator. Again, in this labelling task it is not clear how to define such a signal, and the corpus is not applied.

Therefore, semi-supervised learning could be the best option to breakthrough the bottleneck brought by the size of training data, and via such an approach we can make fully use of labelled corpus to learn how to predict future unlabelled data precisely.

As discussed in Section 2.6, an obvious benefit of co-training technique is users are able to control the learning progress, e.g. stop it when the performance reaches an acceptable point (usually at a peak point after which the performance declines). Hopefully, it will produce more high standard labelling data without human intervention.

4.1 The Co-training Algorithm on CRF

As previous results have shown that CRFs classifier achieved the highest F-scores among all four baselines, I choose to use CRF classifier with best metrics assignments for experimenting the co-training algorithm. Here I implemented co-training algorithm for CRF on both documents and sentences.

The co-training framework for CRFs is similar to the general co-training framework for classification problem in [Nigam and Ghani 2000]:

L: the set of labelled units
U: the set of unlabelled units
C_i: classifier i in co-training
F_i: the feature from which classifier i is constructed

```
1. Read in and parse a small set of labelled units $L$;
2. Read in and parse a large set of unlabelled units $U$;
3. Prepare testing data;
4. Design two sets of features $F_1$ and $F_2$ with the goal to achieve maximum conditional
independences of each other;
5. Instantiate two CRF classifiers $C_1$ and $C_2$ with $F_1$ and $F_2$ respectively;
6. Loop for $N$ iterations:
    - For each classifier $C_i$:
        - Train classifier $C_i$ from labelled document set $L$;
        - Use trained classifier $C_i$ to label all units from $U$;
        - Sort those labelled units by confidence values from high to low;
        - Select the top $K$ results which satisfy thresholds;
        - Remove selected units from $U$ and add them (with predicted labels) to $L$
```

4.2 Methodology

4.2.1 Co-training Models

To achieve more obvious improvements in co-training, the feature sets used by two classifiers should be as largely conditional independent as possible. In this project I implemented two groups CRF models with different feature sets.

Firstly, a straight-forward option is the combination of below two CRFs which are literally defined (see Figure 4.1):
(1) a 'tokens model': this CRF classifier has information about tokens only;

(2) a 'part-of-speech tags model': the second one only knows what sorts of POS tags are labelled with different TimeML tags.

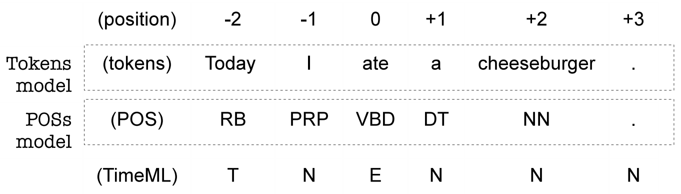


Figure 4.1: The illustration of 'tokens model' and 'part-of-speech tags model'.

Another more complex combination is (see Figure 4.2):
(1) a 'target position model': it's only features are token and POS tag for the token being classified;

(2) a 'context model': the only features are tokens and POS tags surrounding the token being classified within a specific range.

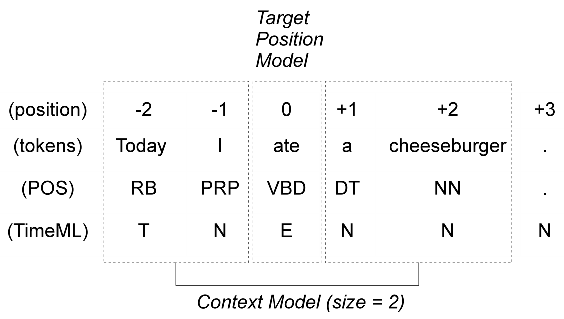


Figure 4.2: The illustration of 'target position model' and 'context model' where the window size is 2.

4.2.2 Thresholds Selection

Unlike supervised learning, unsupervised learning algorithms have difficulties to make correct judgements about their learning qualities. One of the key points of devised co-training framework is how to select the best results made by each classifier. Ideally, from all data labelled by classifier C_i in each iteration, the selected results shall be the most 'informative' ones to the other, so that one classifier can learn from the counterpart's predictions to compensate its own limitations. Therefore, the outputs chosen by the co-training algorithm shall not only have high conditional confidences, but also need to satisfy the minimum ratios of *EVENT* and *TIMEX3* tags.

Firstly, in CRF++ outputs, a sub-total of conditional confidence is given for each training unit (if the training unit is a document, then a confidence value of that document is calculated; or if the training unit is a sentence, then each sentence can be compared with its this value too). So with values of conditional confidence I can sort prediction results in a descending order.

Next, starting from the top of the sorted list (of higher conditional confidence), the results are further filtered by *EVENT* and *TIMEX3* ratio thresholds. Exploring optimal thresholds can be a difficult task and involve a great amount of empirical efforts. Below are some suggestions of thresholds that I have concluded from experiments:

- Threshold *CC*: the conditional confidence of a training unit u shall be greater than a constant C_1 $(C_1 \geq 0)$;

- Threshold *E*: the ratio of tokens labelled with *EVENT* tag in that training unit shall exceed a minimal value C_2 $(C_2 \geq 0)$;

- Threshold *T*: the ratio of tokens labelled with tag *TIMEX3* in the training unit shall exceed a minimal value C_3 $(C_3 \geq 0)$;

- Threshold *ETCC*: conjunction of (a), (b) and (c);

- Threshold *E+T*: the sum of 'the ratio of tokens labelled with tag *EVENT*' and 'the ratio of tokens labelled with tag *TIMEX3*' in the training unit shall exceed $(C_2 + C_3)$.

Here is a short explanation of how to calculate the minimal values of Thresholds *E* and *T*. For all training data, I count the percentages of tokens which are labelled with *EVENT* or *TIMEX3* out of all tokens in each training unit (a document or a sentence), and then calculate the mean percentage with standard deviations. Finally, the values of more 'relaxed' thresholds are calculated by subtracting standard deviations from the mean percentages in order to tolerate some slightly low ratios. In this task, Threshold $E = 0.0936$, and Threshold $T = 0.0153$. Practical results favour the Threshold *ETCC* with $C_1 = 0$ as the co-training algorithm generates more desirable outputs with this threshold than with others.

Third, after filtered by Threshold *ETCC*, the top *K* results are more likely to contain a certain number of good predictions on all *EVENT*, *TIMEX3* and *NONE* tags, therefore it is more reliable to add them to the labelled data set *L*.

Note there could be more combinations of above criterion, or one can devise different thresholds. In fact, there exists possibility that applying better thresholds may increase the performance of co-training algorithm slightly. Nevertheless, this improvement cannot be significantly greater than current results as proved in the section 4.4.

4.2.3 Predict the Maximum Potential of Co-training

The potential of the co-training algorithm can be predicted by varying the amount of training data to CRFs, to see how much additionally accurately labelled training data will help in practice. This step has nothing to do with co-training but just a simple CRF training with different fractions of the training data set. More specifically, this is done by randomly dividing the training data into five parts. Start training from any part and then increment training data by 20% (33) at one time, i.e. to 40% (65), 60% (97), 80% (131) until all 163 files. The testing data remains unchanged. Again, *K*-fold cross validation is applied to each test to ensure the stability of testing results.

This step is important for establishing before running further experiments as I will have a general idea about to what extent I can improve co-training results with more data. For example, if there is little difference between 80% to 100%, then maybe there is not much room for improvement beyond this difference. I imagine that the curves of co-training could experience some increases in the first few iterations and reach a peak point somewhere, but later as more and more predicted outputs (they are not pure samples but contain noise) are added into the labelled data set *L*, there shall be downward trends.

The results are shown in tables 4.1 and 4.2 and corresponding figures. For both cases of CRFs training, there are obvious improvements in F-scores when the training data size grows from 20% to 60%. But there are no significant changes from 60% of data to the whole size of training dataset (only approximately 1.9% increase). Thus, the improvement of co-training algorithm by increasing training data size shall be limited, and in numerical this value is about 2%.

Number of documents for training	Precision	Recall	Accuracy	F-score
33	88.48%	65.31%	92.64%	0.7401
65	90.29%	75.32%	94.53%	0.8145
97	90.72%	77.58%	94.96%	0.8306
131	90.86%	78.28%	95.07%	0.8350
163	90.99%	80.06%	95.34%	0.8467

Table 4.1: Average performance of CRF classifier training on documents with various amount of training data, which indicates the maximum possible improvements in F-scores by increasing training data from 60% to 100% is about 2% only.

Number of sentences for training	Precision	Recall	Accuracy	F-score
33	88.73%	65.99%	92.79%	0.7458
65	90.21%	75.88%	94.60%	0.8180
97	90.62%	77.68%	94.95%	0.8307
131	90.74%	78.71%	95.13%	0.8374
163	90.91%	80.13%	95.35%	0.8467

Table 4.2: Average performance of CRF classifier training on sentences with various amount of training data, which indicates the maximum possible improvements in F-scores by increasing training data from 60% to 100% is about 2% only.

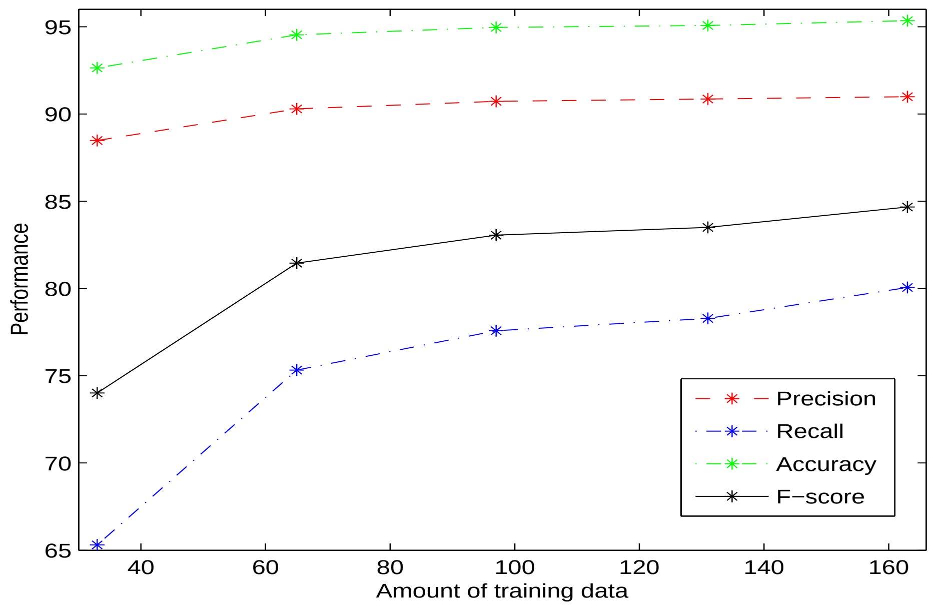


Figure 4.3: The comparison of varying amount of training data

4.3 New Dataset

The unlabelled document set U contains 2,231 news articles from the websites of the Australian and the Sydney Morning Herald. Those reports covered a variety of topics from 'nuclear energy', 'Japan earthquake', 'health' to 'Afghanistan' etc. Those documents are extracted from web pages, during that process only key text contents are preserved and irrelevant labels are ripped off. Further discussion of extracting key text contents will beyond the scope of this thesis.

4.4 Results

4.4.1 Co-Training on Documents

First of all, let's have a close look at the output format of CRF++. Table 4.3 shows the beginning part of a CRF++ output on testing data. The verbose level is set to 1 in this example.

```
# 0.000007
   The       DT    <NONE>/0.988620
 primary     JJ    <NONE>/0.931491
   vote      NN    <EVENT>/0.568387
    of       IN    <NONE>/0.996568
   both      DT    <NONE>/0.923953
   the       DT    <NONE>/0.995167
  Labor      NNP   <NONE>/0.951198
  Party      NNP   <NONE>/0.933114
   and       CC    <NONE>/0.994960
   the       DT    <NONE>/0.997772
Coalition    NNP   <NONE>/0.963055
   have      VBP   <NONE>/0.998275
   been      VBN   <NONE>/0.982462
   ...       ...         ...
```

Table 4.3: An example of the CRF++ output

The first and second column, from left to right, are tokens and their part-of-speech tags . The third column shows predicted TimeML tags with marginal probabilities. And the first line "# 0.000007" shows the conditional probability for the testing unit.

One may notice that in this example, though the marginal probabilities of each output tag are mostly above 0.90, however the conditional probability of the entire prediction is very low as only $7 * 10^{-6}$. Actually, the length (number of all tokens) of this document is 148, which is a very short article already in comparison with others in the corpus.

Also, according to observation, the conditional likelihood of a training unit calculated by CRF++ is very close to the product of those marginal probabilities at each token. As a power 100 of 0.9 is in 10^{-5}, it is not surprised to see most of predictions on long documents are near zero confidence values. As a result, the top K results selected usually fail to introduce solid predictions to the labelled data set L.And that leads to an ugly fact that documents of shorter length are more favoured by CRF++. Clearly, they are not the ideal results that I am looking for.

To eliminate the bias due to the length of document, the conditional likelihood C of a document of length k is normalised by kth root,

$$C^* = C^{1/k} \tag{4.1}$$

where $C*$ is the normalised conditional likelihood.

Figure 4.4, 4.5 and 4.6 are results of co-training on documents with CRF models for 25 iterations where the F-scores are scaled to to comparable. At each iteration, only the top 3 documents with higher normalised conditional likelihood values and satisfying Threshold *ETCC* are added to the labelled set L, as adding too many documents at a time could introduce more noises than good examples.

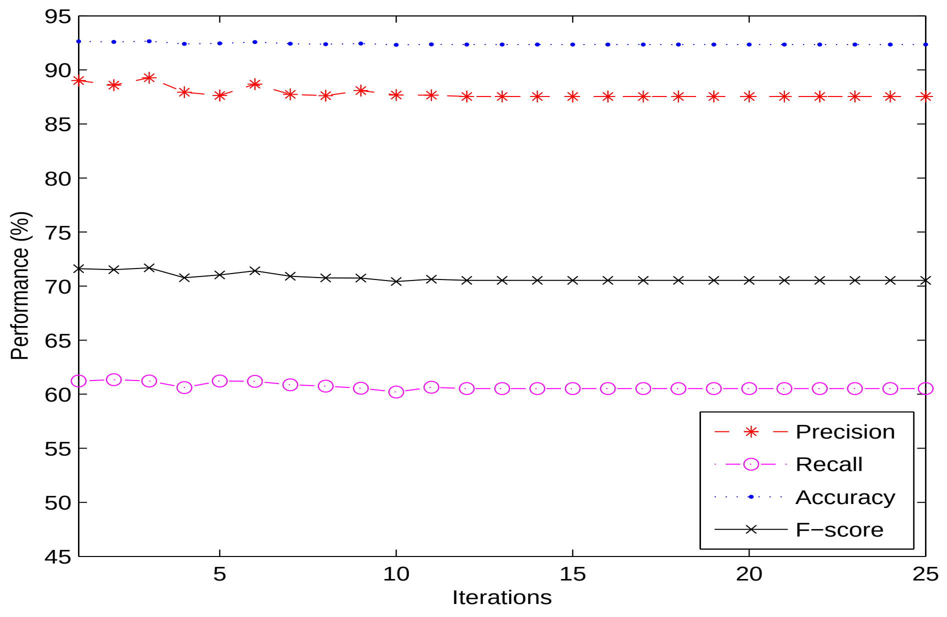


Figure 4.4: The performance of the 'tokens model' by co-training on documents, where no obvious changes can be captured.

The experiment results show that though the probabilities are normalised, there is no obvious improvement in performance. Thus, I propose **applying co-training on sentences** instead of documents will be more effective in this task, as discussed and experimented in next section.

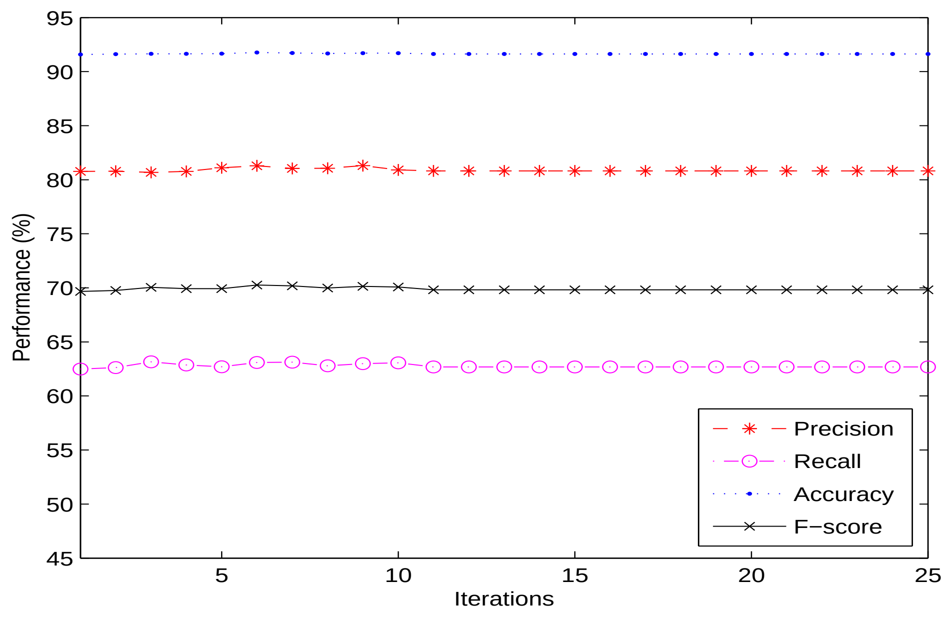


Figure 4.5: The performance of the 'POS tags model' by co-training on documents, where no obvious changes can be captured.

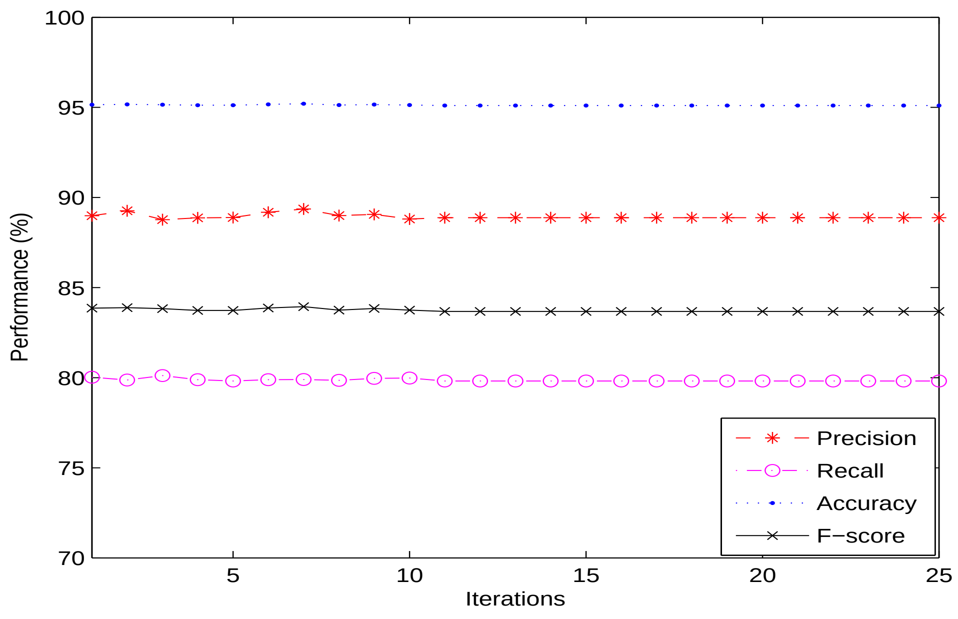


Figure 4.6: The performance of a combined model of the 'tokens model' and the 'POS tags model' by co-training on documents, where no obvious changes can be captured.

4.4.2 Co-Training on Sentences

With the aid of sentence breaking technique, all 2,231 unlabelled documents are broken down into a pool of individual sentences. So far the size of unlabelled sentences set *U* is up to 58,732. However, a drawback of co-training on sentences is that the CRF may lose the context of the overall document. But as CRF++ uses linear chain CRFs, and window size adopted is 1, it is safe to ignore the drawback.

The first group of models (classifiers), as discussed previously in section 4.2, is the combination of a 'tokens' model and a 'part-of-speech tags' model. And the second group is a combination of a 'target position model' and a 'context model'. Each model has only information about its own features but has no knowledge about features of the other one.

Similarly, the conditional likelihoods of the output tags of each sentence are normalised according to the length of that sentence. A maximum 100 sentences which are selected from top results and satisfy Threshold *ETCC* can be added to the labelled sentences set *L* and removed from *U*. F-scores are scaled in the following figures.

Figure 4.7 shows the change of performance of the 'tokens' classifier. The initial F-score of this classifier is 0.7157, during the co-training phrase, the 'tokens model' manages to reach a maximum F-score of 0.7392 at iteration 9. After that, as training continuing and more sentences are added into the labelled set 'L', the curve of F-scores gradually drops down. Moreover, the graph illustrates that the decline in F-score is due to the curve of precision going downward faster than the slightly increase in the curve of recall after iteration 9.

From the results in Figure 4.8, however, it seems that the 'tokens model' fails to train the 'part-of-speech tags model' to perform better during co-training. The downward trend of F-score is very obvious, as a result of rapidly decreasing in precision. Also, the curve of accuracy has experienced a slight decline from the beginning until the last iteration. As a comparison sample, the co-training results of a combined model are given in Figure 4.9.

The reasons behind those curves are: first, there exists a pattern in TimeML annotation such as tokens with POS tags like `VB`, `VBD` and `VBN` are labelled with *EVENT* more frequently than other tokens. With this experience, the first model 'tokens' tends to become more confident in making correct predictions.

On the other hand, the unsatisfied results in the 'POS tags model' are expected, as there is only a limited amount of POS tags (in comparison of a huge amount of tokens (words) in linguistic world of human languages). In other words, many tokens which are labelled with *EVENT* or *TIMEX3* in a certain context may not be assigned the same tags again in other circumstances. In addition, as new words are continuously

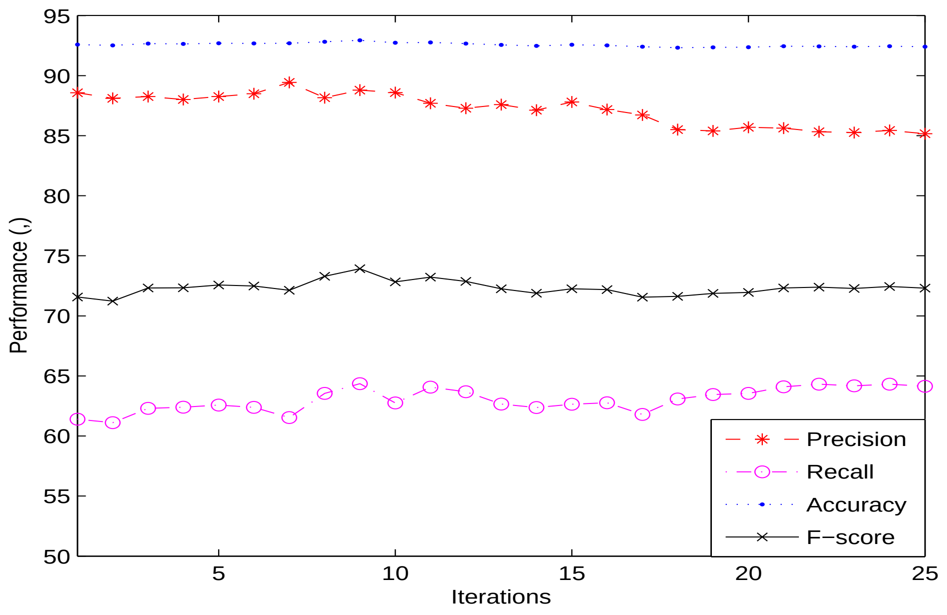


Figure 4.7: The performance of the 'tokens model' by co-training on sentences, where the model reaches the highest F-score of 0.7392 at iteration 9.

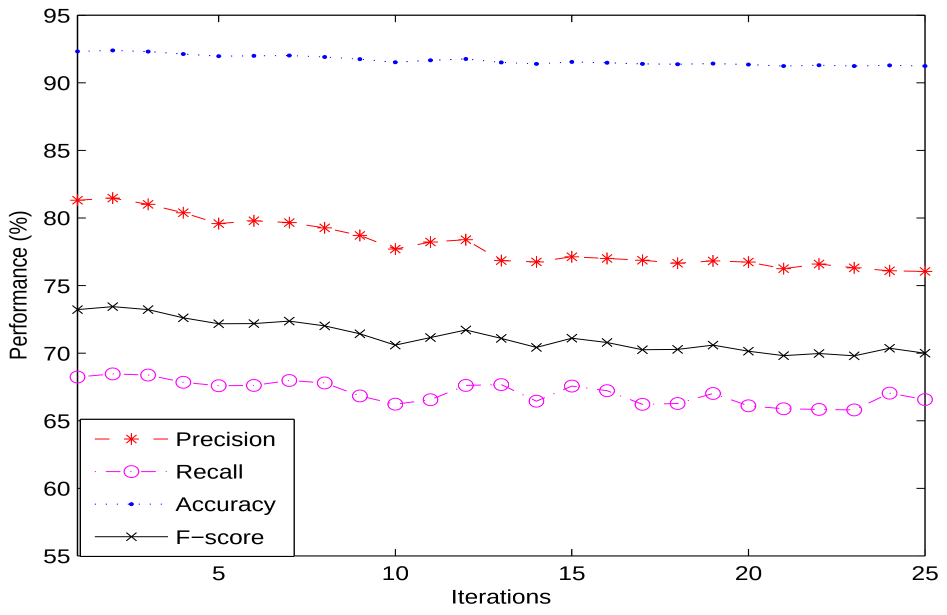


Figure 4.8: The performance of the 'POS tags model' by co-training on sentences, which indicates that this model fails to benefit from co-training with the 'tokens model'.

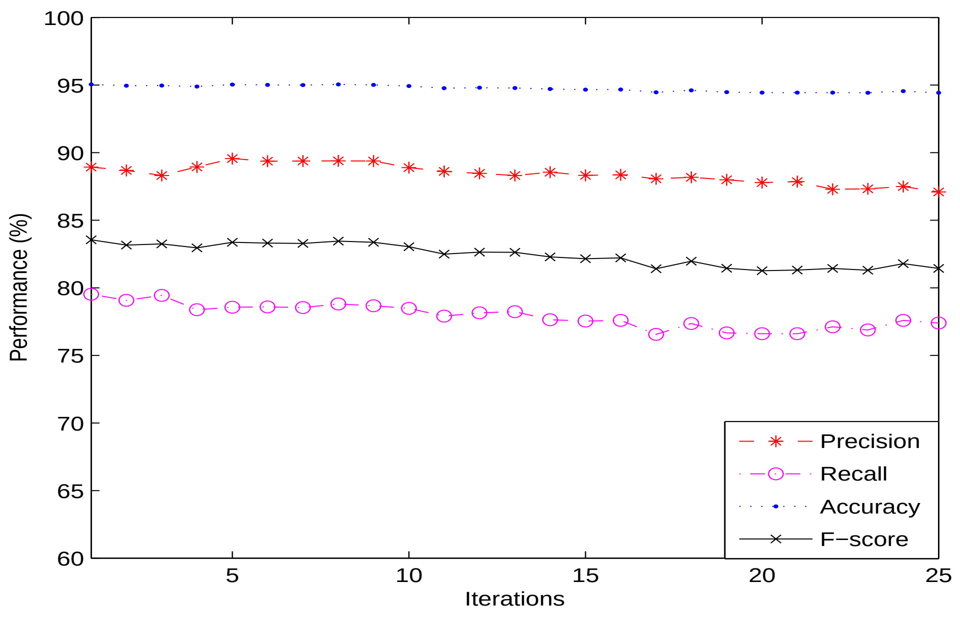


Figure 4.9: The performance of a combined model of the 'tokens model' and the 'POS tags model' by co-training on sentences, which performances is suffering from the failure in the 'POS tags model'.

added into the labelled sentence set 'L', it turns out to be very difficult for the 'POS tags model' to discover a useful pattern from tokens, and that leads to the decreases in the performances of both this model and the combined model during co-training.

Now it's time to turn to look at the second group of CRF classifiers for co-training. The 'context model' in the second group works by assigning it a specific window size, to define the range of context being considered when doing classification. Conventionally, window size 0 means only features at current position of prediction are considered, and this is just the case of the 'target position model'.

The co-training results of window size 1 to 4 of the 'context model' and the 'target position model' are presented and analysed below in four independent parts.

Firstly, **when window size is 1**, the F-score of the 'target position model' shows a downward trend from the beginning of training. It may indicate in co-training, the 'context model' with a limited scope of surrounding context makes few contribution to the 'target position model', therefore the later model learns nothing useful to boost its performance. Apparently, there is a need to expand the size of the sliding window later.

The 'context model' has experienced a slightly increase in F-score during co-training:

starting from 0.6423, it finally achieves the highest F-score 0.6259 at the 20th iteration. Specifically, there is a decrease in precision while a greater increase is observed from the curve of recall. This result suggests the performance of the 'context model' is improved by the outputs of the 'target position model'.

The combined model of previous two shows less observable changes but there is a minor 0.0049 increase in F-score.

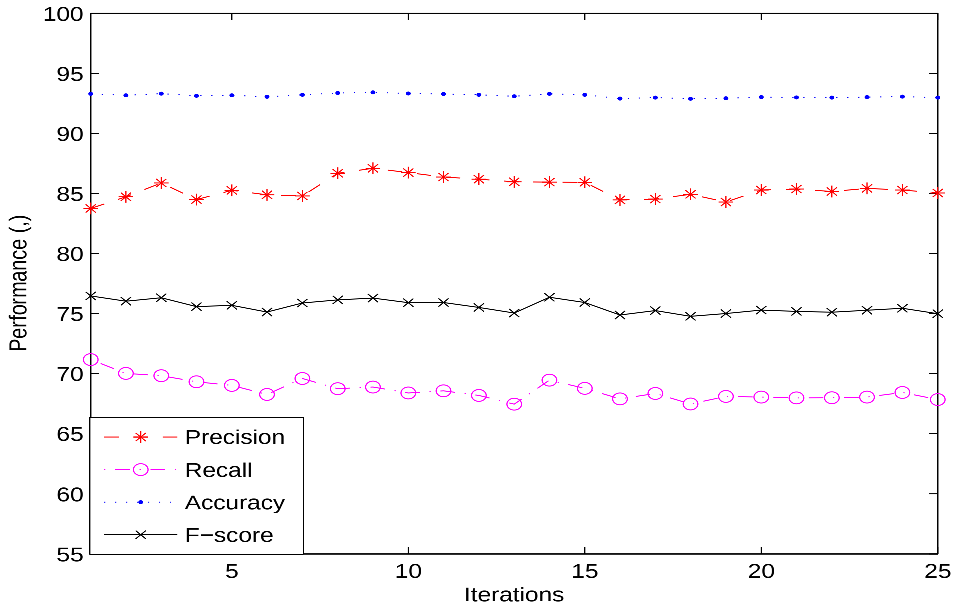


Figure 4.10: The performance of the 'target position model' by co-training on sentences with the 'context model' when window size = 1. It shows with limited scope of surrounding context its counterpart 'context model' fails to train this model well.

In contrast, **when window size is increased to 2**, which means tokens and POS tags of the previous two positions and next two positions are also included (position indexes: $-2, -1, +1, +2$) in the prediction of TimeML label of current position, the F-score of the 'target position model' starts improving from 0.7639 to the highest 0.7789 at the 14th iteration. After that the curve gradually goes down (Figure 4.13). This means with more context information, the outputs produced by the 'context model' are now beneficial to this model in co-training.

The improvement in the 'context model' during co-training is slow but steady. Its F-score reaches the highest 0.6571 at the 23rd iteration, which is also a 0.0143 growth in comparison with the initial 0.6428 (Figure 4.14). The performance of the combined model is shown in Figure 4.15.

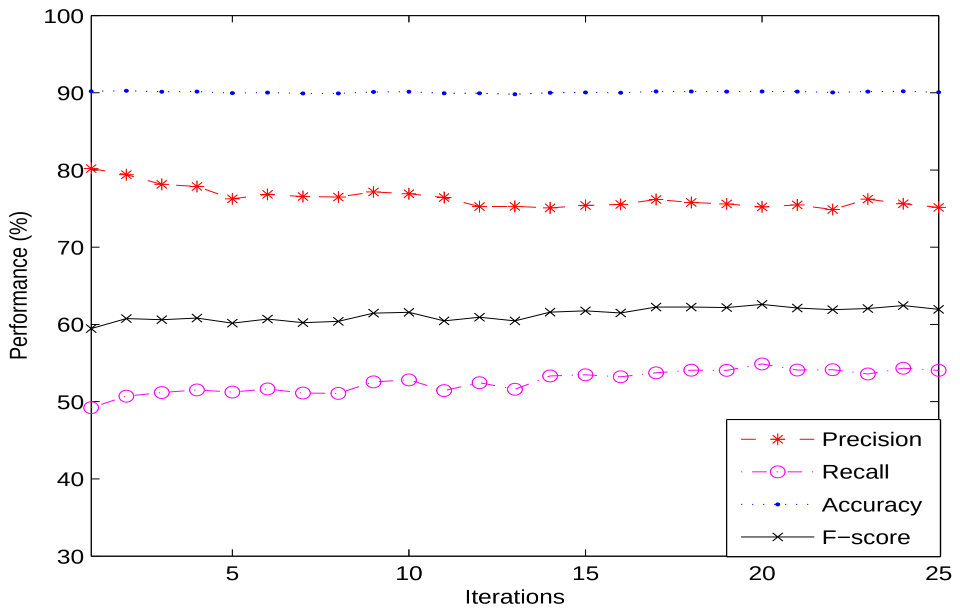


Figure 4.11: The performance of the 'context model' by co-training on sentences with the 'target position model' when window size = 1, the highest F-score 0.6259 at the 20th iteration.

When window size is 3, similarly the F-score of 'the target position model' reaches the highest 0.7717 at the 13th iteration, but it is only a minor change of 0.0078 improvement than initial value (Figure 4.16). However, in comparison with the results of window size 2, where the curve of recall experiences a climb before decline, there is no any rise of recall during co-training. At this point, I make an assumption that window size 2 could be the best option for co-training the combination of the 'target position model' and the 'context model'.

And for the 'context model', the maximum F-score 0.6665 is on the 25th iteration (see Figure 4.17).

At last, **the window size is further increased to 4**. The maximum F-score 0.7728 in the 'target position model' is obtained at iteration 7 and then the curve declines (Figure 4.19), which is similar to the change in window size 3 but the value is still smaller than the increase in window size 2. This further justifies my assumption that in co-training of these two CRF models, window size 2 may be the best option to have the 'target position model' learning more from the 'context model'. In Figure 4.20 the maximum F-score 0.6335 is obtained at iteration 17.

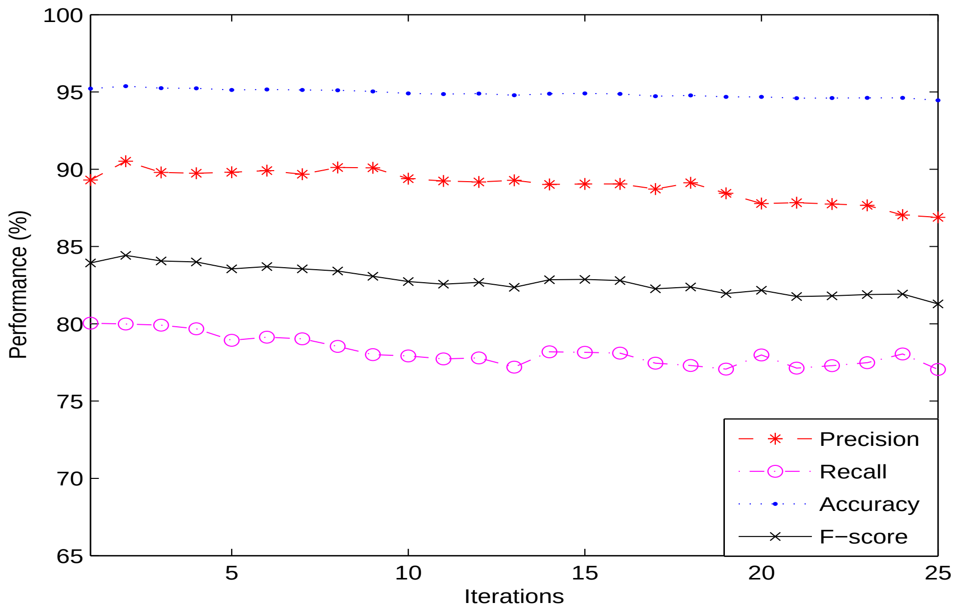


Figure 4.12: The performance of a combined model of the 'target position model' and the 'context model' by co-training on sentences when window size = 1, with a minor 0.0049 increase in F-score because of the failure of the 'target position model'.

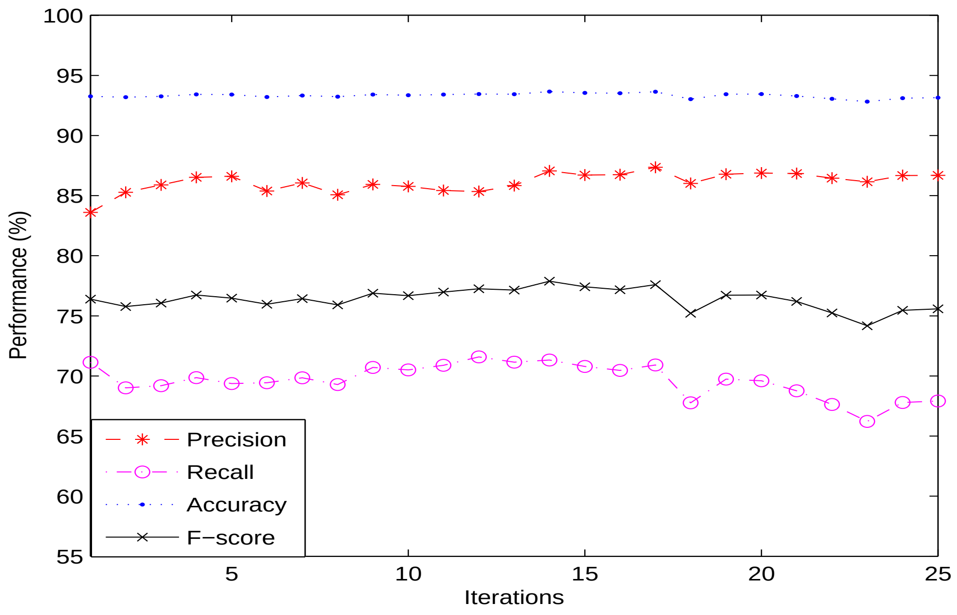


Figure 4.13: The performance of the 'target position model' by co-training on sentences with the 'context model' when window size = 2, a noticeable increase of F-score at iteration 14 is 0.0150 higher than beginning value.

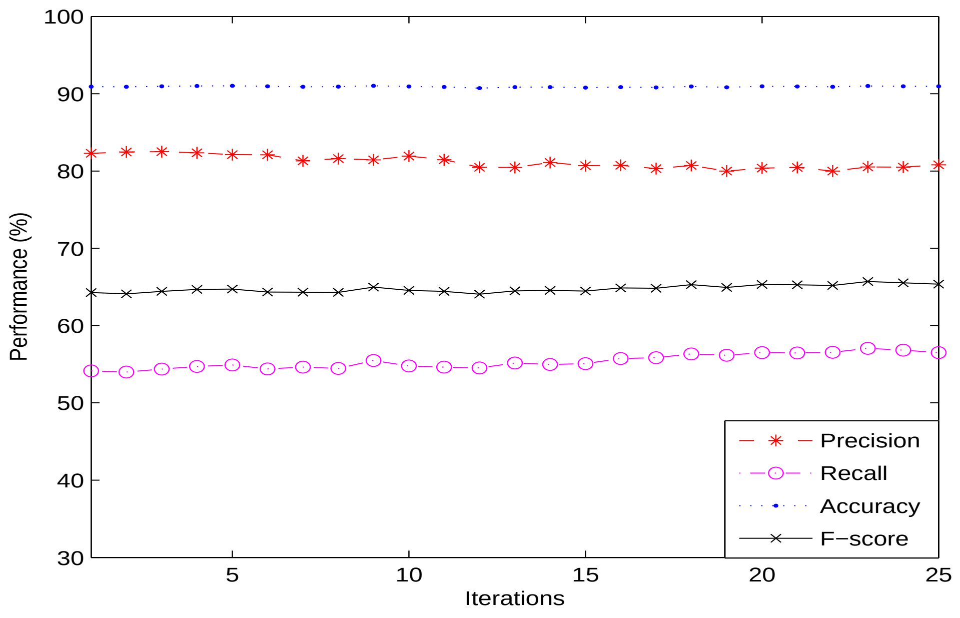


Figure 4.14: The performance of the 'context model' by co-training on sentences with the 'target position model' when window size = 2, where it reaches the highest 0.6571 F-score at iteration 23.

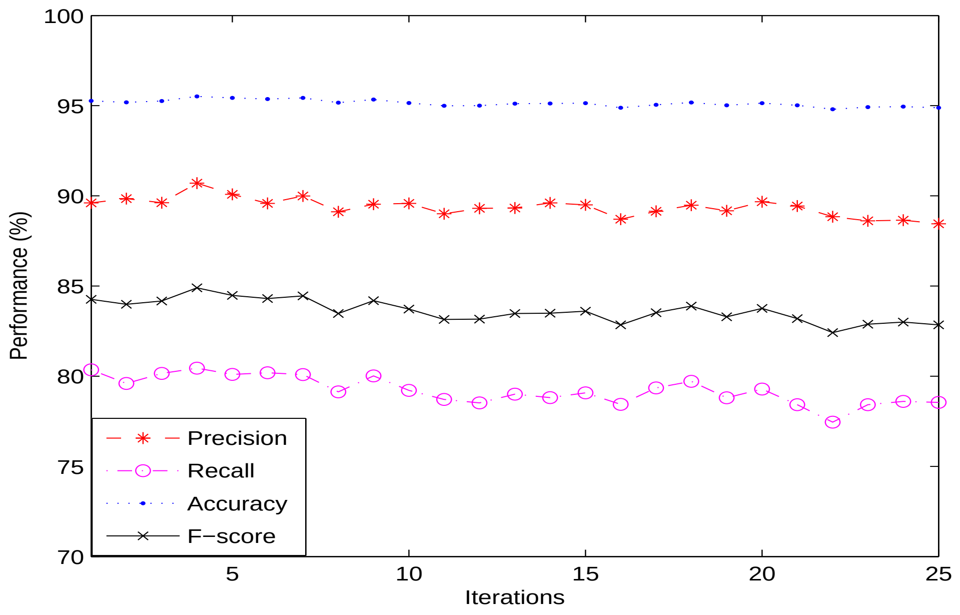


Figure 4.15: The performance of a combined model of the 'target position model' and the 'context model' by co-training on sentences when window size = 2. It combines the features of both models and look more steady. Overall, the F-score has a 0.0064 increase at the peak.

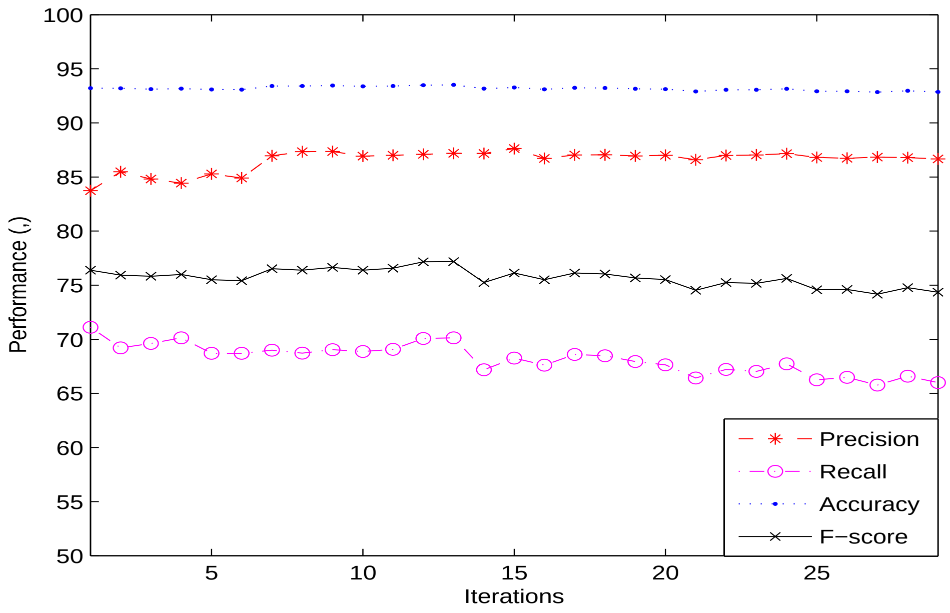


Figure 4.16: The performance of the 'target position model' by co-training on sentences with the 'context model' when window size = 3, a lower rise in F-score appears at iteration 13 than window size 2. It may indicate when window size is increased 3, too-much context information with noises bring less contribution to the performance of this model.

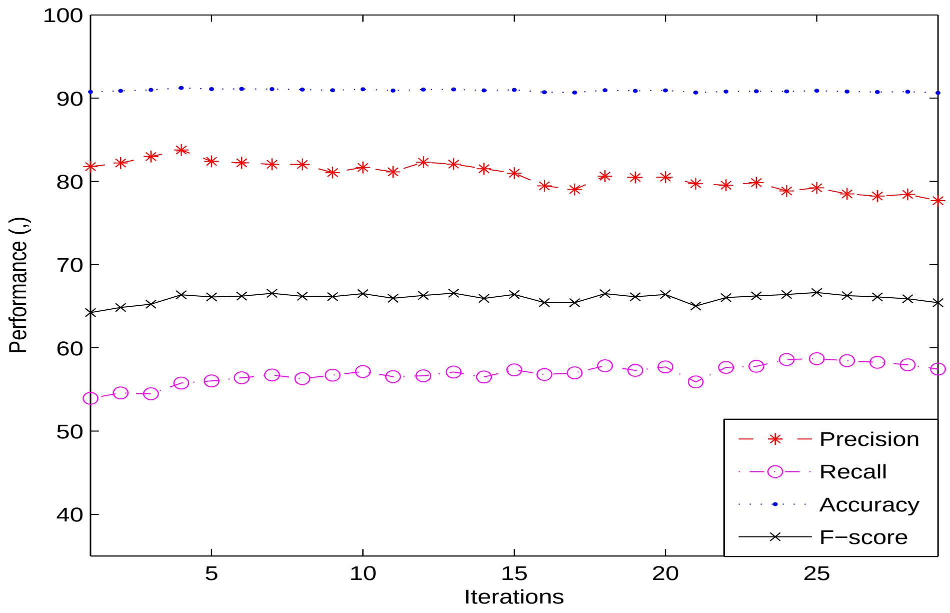


Figure 4.17: The performance of the 'context model' by co-training on sentences with the 'target position model' when window size = 3, the highest F-score 0.6665 is on the 25th iteration.

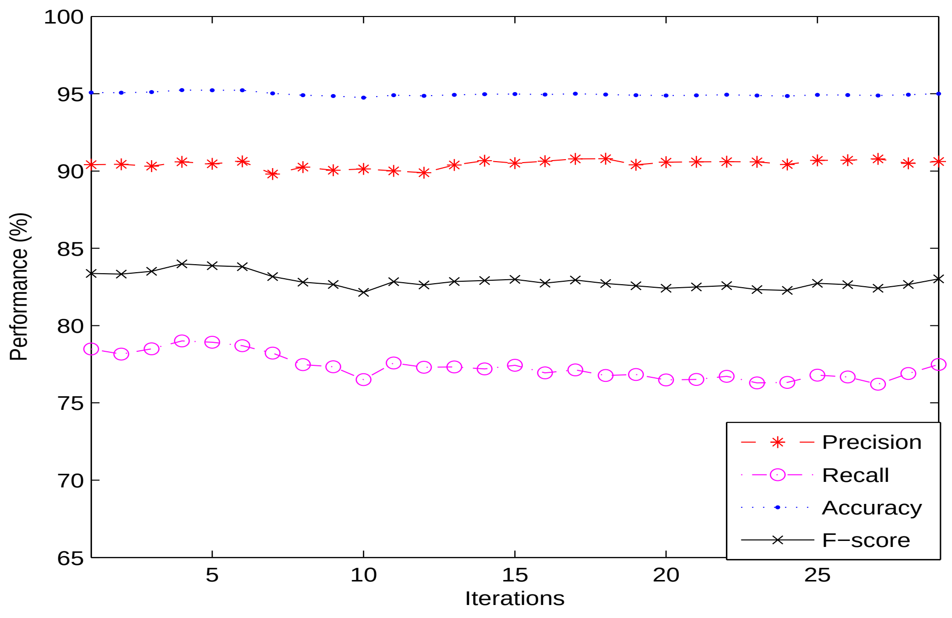


Figure 4.18: The performance of a combined model of the 'target position model' and the 'context model' by co-training on sentences when window size = 3. There is a 0.0062 rise in F-score at the highest point.

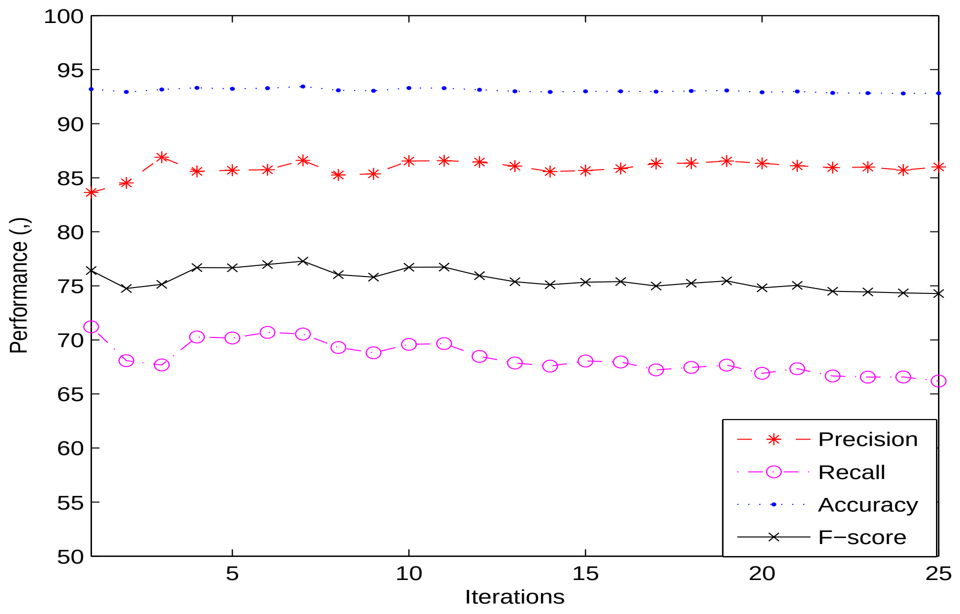


Figure 4.19: The performance of the 'target position model' by co-training on sentences with the 'context model' when window size = 4. The maximum F-score 0.7728 is obtained at iteration 7 but then the curve declines.

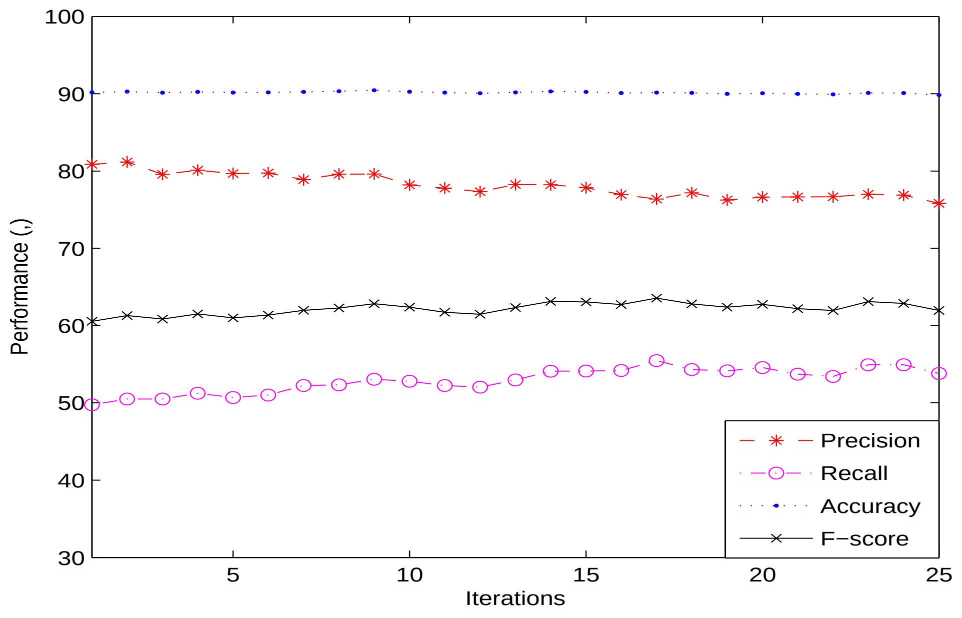


Figure 4.20: The performance of the 'context model' by co-training on sentences with the 'target position model' when window size = 4. The maximum F-score 0.6335 is obtained at iteration 17

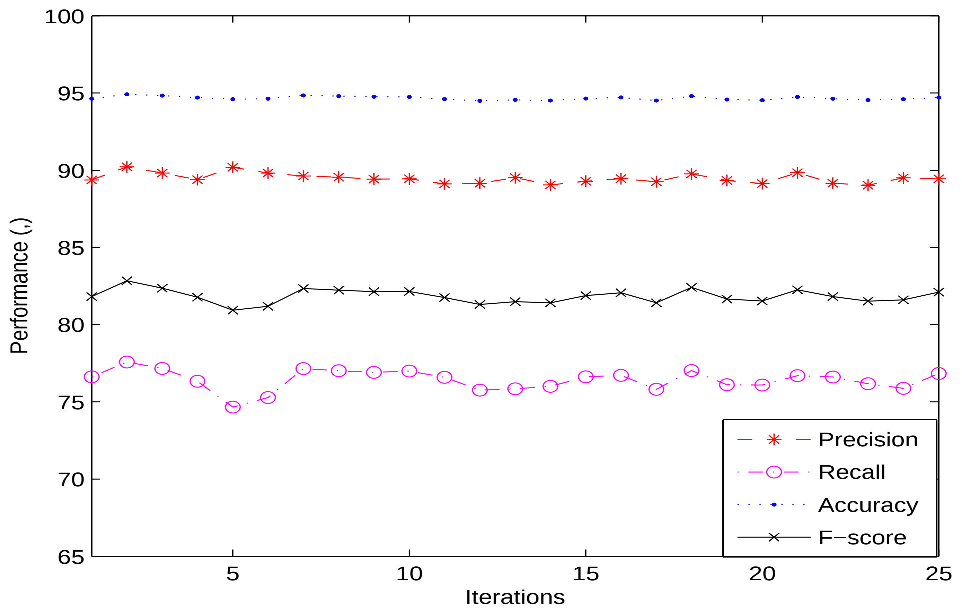


Figure 4.21: The performance of a combined model of the 'target position model' and the 'context model' by co-training on sentences when window size = 4. Yhere is a 0.0102 growth at the highest point.

4.4.3 Analysis of Co-Training Results on Sentences

A dozen of figures of the co-training results on sentences need to be concluded and analysed in a cleaner manner in this separate section to capture the insights more effectively.

Table 4.4 concludes all important statistics of those figures. The values in the table are the highest F-scores achieved of each model during co-training. The numbers in the parentheses represent the changes between the highest F-scores and initial values of each run. Moreover, another four graphs illustrate those changes from another prospective.

Window Size	Target Position Model	Context Model	Combined Model
1	0.7648 (+0.0000)	0.6259 (+0.0312)	0.8443 (+0.0049)
2	0.7789 (+0.0150)	0.6571 (+0.0143)	0.8490 (+0.0064)
3	0.7717 (+0.0078)	0.6665 (+0.0234)	0.8399 (+0.0062)
4	0.7728 (+0.0086)	0.6355 (+0.0301)	0.8284 (+0.0102)

Table 4.4: Comparisons of F-scores and their change rates of various window sizes of the combination of the target position model and the context model in co-training

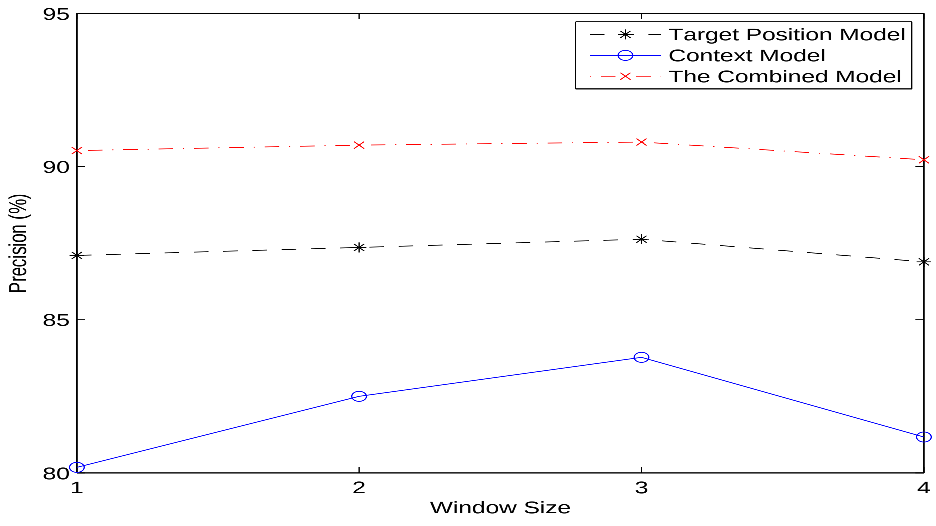


Figure 4.22: The comparisons of 'target position model', 'context model' and the combined model of both with various window sizes in terms of precision. The models achieve maximum precision at window size 3.

According to those statistics and charts, I have concluded several observations:

- The Context Model has been benefited more from the Target Position Model rather than the other way around;

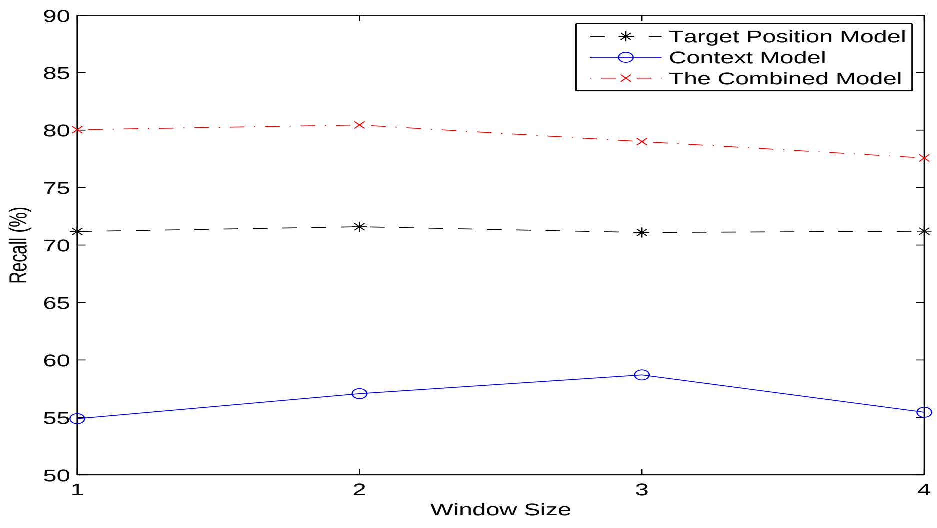


Figure 4.23: The comparisons of 'target position model', 'context model' and the combined model of both with various window sizes in terms of recall. Though the 'context model' reaches the maximum value at window size 3, but the 'target position model' and the combined model favour window size 2 statistically.

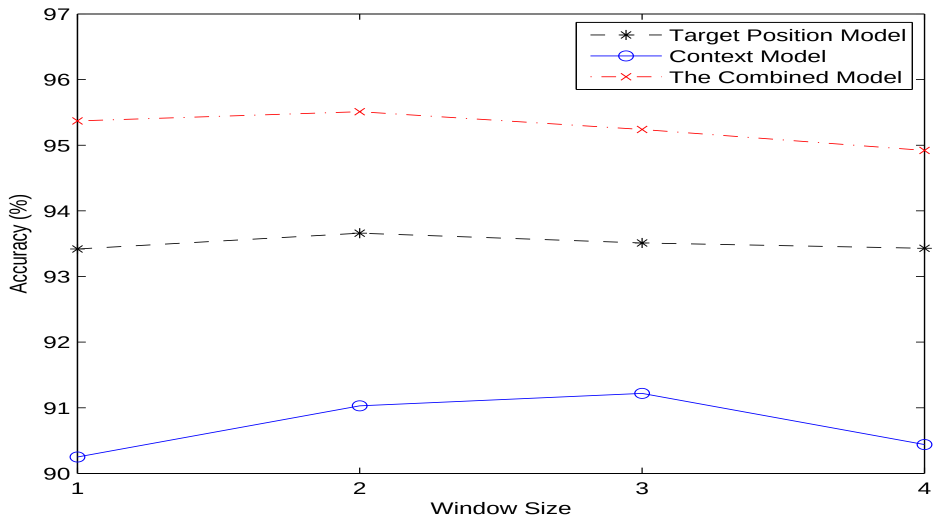


Figure 4.24: The comparisons of 'target position model', 'context model' and the combined model of both with various window sizes in terms of accuracy. Though the 'context model' reaches the maximum value at window size 3, but the 'target position model' and the combined model favour window size 2 statistically.

- The highest F-scores of the Target Position Model and combined model are both observed at window size 2, but the maximum F-score of the Context Model is at

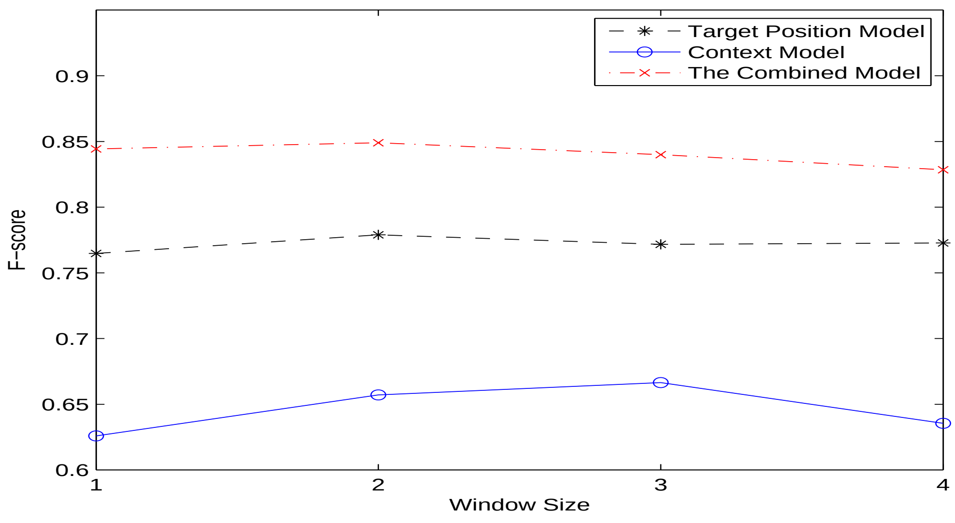


Figure 4.25: The comparisons of 'target position model', 'context model' and the combined model of both with various window sizes in terms of F-score. Though the 'context model' reaches the maximum value at window size 3, but the 'target position model' and the combined model favour window size 2 statistically.

window size 3;

- As discussed in previous section, it appears that the best option of window size to co-training with those two CRFs on sentences is 2, as the increase of the F-score of the Target Position Model is almost double (+0.0150) than other cases. Though the improvement in the Context Model is the least significant, but more balanced improvements (around 2%) on both models are observed.

As the peak performances of both CRFs are normally not achieved at the same iteration, it is almost impossible to find a perfect labelled data set which can maximise the F-scores of both classifiers. To evaluate the labelled data generated by co-training algorithm, I decide to extract the labelled data set at the 14th iteration with window size 2, where the Target Position Model reaches the highest 0.7789 F-score. Apparently, one can select other labelled data set from peak performance of the other classifier or even different window sizes, the labelled data set I selected is just one option of many which I think it may be able to produce more interesting information.

4.5 Evaluate Co-training Results on Baseline Classifiers

One way to prove that new generated labelled data by co-training algorithm are useful is adding them to original training data set to re-train those baseline classifiers. Originally, there are 163 documents as training data and another 18 documents served as testing data. My method is keeping the testing data unchanged, but mixing the new labelled data set with the old training data (the overlapped parts are removed as original 163 training documents are counted twice). In such a way, because testing on the same groups of data as before, the results will be more persuasive if they are higher than previous results. To ensure the stability of the results, I use 10-fold cross validation like what I did previously.

The baseline classifier selected to make this comparison is the Naive Bayes model, which has achieved a 0.7250 F-score with original training/testing data. If the Naive Bayes model can perform better than what it did before, then other baselines are very likely to reach the same conclusion.

Classifiers	Precision (%)	Recall (%)	Accuracy (%)	F-score
NB (old)	63.96 (± 4.17)	85.66 (± 1.43)	89.59 (± 1.20)	0.7250 (± 0.0333)
NB (new)	76.84 (± 7.54)	87.26 (± 3.90)	93.50 (± 2.11)	0.8115 (± 0.0622)

Table 4.5: Comparison of Naive Bayes classifier with original dataset and new dataset improved by co-training results

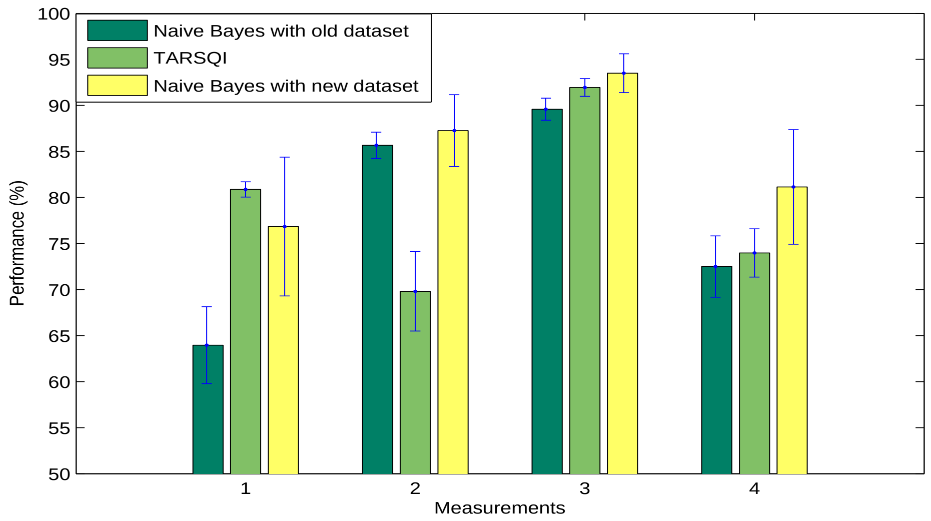


Figure 4.26: The Naive Bayes classifier outperforms its originally performance with the new labelled data set, where four numbers from 1 to 4 represent precision, recall, accuracy and F-scores

In comparison with previous performance, there are noticeable improvements for the Naive Bayes classifier with the new labelled data set from co-training framework. Even though the standard errors of four measurements of the new training data set become larger too, however, generally speaking the Naive Bayes classifier makes a better prediction with the same testing data than it previously did because of higher means on four measurements. This results suggest that the new labelled dataset from the proposed co-training framework bringing positive improvements to the supervised learning algorithms in this classification task.

4.6 Summary

In this section, I introduced a framework of applying co-training algorithm to train two independent CRF models to lean from each other, and implemented with both documents training and sentences training. By co-training on a large set of unlabelled documents, there are about 2 % improvements on F-scores of two groups of CRF models. And at the end of this section, the experimental results show that with the help of new labelled data from co-training, the Naive Bayes classifier makes more accurate prediction on the same testing data.

Chapter 5

Conclusion

5.1 Summary of achievements

To sum up, below I list my contributions in addressing this time and events extracting task, and in applying co-training framework in sequential text data from an innovative perspective:

- As a supervised learning algorithm, the **Conditional Random Fields (CRFs)** beats the 'state-of-the-art' toolkit TARSQI in this field with significant advantages in terms of precision, recall, accuracy and F-score. (see Section 3.2)
- Bottleneck for better performances of supervised learning algorithms appears to be limited size of training data. But it is time-consuming and costly to manually label new corpus of satisfaction constantly. To address this bottleneck, I proposed **a framework of Co-Training** via which I can exploit a large amount of unlabelled data in training. (see Section 4.1 and 4.2)
- I observed severe deficiencies in standard co-training algorithm when applied to sequential data which need to be addressed with novel solutions:
  - Most major observation regards the **selection of labelled examples** in co-training. In sequential models, most confident results often are the data without any time or event labels - but these new examples do not help identify new cases of time or events. There exists a need to modify co-training selection criterion to also include informativeness of the example in terms of the number of event and time labels. With this and training on examples as sentences, the proposed co-training framework shows marginal improvements over the non-co-trained baseline. (see Section 4.5)
  - The confidence score (likelihood) in sequential models **must be normalised** in terms of the training unit's length, which is not an issue in non-sequential models. (see Section 4.4.1)
  - The CRF model pair works best is a combination of a **'target position model'** (only information of the position to be predicted is visible to this model) and a **'context model' of window size 2** (it can access the context within a

scope of $\pm$ 2 from the position to be predicted but cannot access any information about that position). The experimental results show that with this model pair, both CRFs are benefit from the other's outputs and have made better predictions during co-training than trained separately. (see Section 4.4.2 and 4.4.3)

- Based on an analysis of the impact of the amount of training data on performance and extrapolating these results, it appears that the best co-training algorithm is **approaching the empirical limits of the performance metrics** (see Section 4.2.3). Future performance improvements then will come largely through feature engineering rather than improving learning algorithms.

Simply put, via CRFs and novel insights applying co-training to CRFs, this thesis has produced a fully automated state-of-the-art time and event extraction algorithm that can be used as a critical component in the development of automatic timeline extraction application that motivated this work.

All Java codes of this project are available now at Google Code:

http://code.google.com/p/automated-news-timelines-extraction/

5.2 Future Work

Many efforts could be done to advance my current progress. A few thoughts/suggestions are summarised below with the hope of brining some guidelines and inspirations to future researchers.

- To improve the performance of current Naive Bayes classifier, one can explore additional feature sets. For example, I would like to suggest that the morphological root, e.g. the word 'run' has various forms 'runs', 'ran' and 'running' in different tenses. The reason for this assumption is, as the POS tags have already captured the morphological information such as noun, verb, and adjective, therefore reducing words to a canonical semantic unit like the morphological root 'run' in above example reduces the number of token features and also gives more data per token.

  Some off-shelf tools of this purpose are available without charge, such as WordNet of the Princeton University. New introduced features are preferred to be as simple as those already used in Naive Bayes algorithm.

- I believe there could be a potential to improve current CRF model in CRF++ toolkit, as it is one of the latest techniques and still intensively studied.

- I assume if applying co-training algorithm on unlabelled documents of different genres may not work as good as current results. For example, the languages used in Blogs often contain more informal expressions, slangs, abbreviations and symbols in comparison with the formal English used in news reports. At current stage it is not clear that if co-training on documents of various genres will increase or decrease the performance.

- Another important improvement can be done is using more TimeML tags like 'SLINK' and 'TLINK' which are not considered in this thesis. By studying those *SIGNAL* tags, however, I believe it will be possible to find out the relationships among different events with temporal information, that will be a fascinating but difficult topic.

- Furthermore, to build a mature and robust automated tool, there are many engineering efforts required from software architecture, interface designs, UI design to Web programming. Also it will become necessary to have database to store data in computing when co-training on larger amount of data in the future. Moreover, one can also work to present the timeline in a visually manner, like [Feng and Allan, 2009] did or the SIMILE Widgets described in Section 2.2.

Bibliography

ALLEN, J. 1983. Maintaining knowledge about temporal intervals. *Communications of the ACM 26*, 11, 832–843. (p. 8)

ANDROUTSOPOULOS, I., KOUTSIAS, J., CHANDRINOS, K., PALIOURAS, G., AND SPYROPOULOS, C. 2000. An evaluation of naive bayesian anti-spam filtering. *Arxiv preprint cs/0006013*. (p. 14)

AYTAR, Y., ORHAN, O., AND SHAH, M. 2007. Improving semantic concept detection and retrieval using contextual estimates. In *Multimedia and Expo, 2007 IEEE International Conference on* (2007), pp. 536–539. IEEE. (p. 15)

BISHOP, C. 2006. *Pattern recognition and machine learning*, Volume 4. Springer New York. (p. 10)

BLUM, A. AND MITCHELL, T. 1998. Combining labeled and unlabeled data with co-training. In *Proceedings of the eleventh annual conference on Computational learning theory* (1998), pp. 92–100. ACM. (p. 18)

DUMAIS, S., PLATT, J., HECKERMAN, D., AND SAHAMI, M. 1998. Inductive learning algorithms and representations for text categorization. In *Proceedings of the seventh international conference on Information and knowledge management* (1998), pp. 148–155. ACM. (p. 14)

ESCUDERO, G., MARQUEZ, L., AND RIGAU, G. 2000. Naive bayes and exemplar-based approaches to word sense disambiguation revisited. *Arxiv preprint cs/0007011*. (p. 14)

FREITAG, D. AND MCCALLUM, A. 1999. Information extraction with hmms and shrinkage. In *Proceedings of the AAAI-99 workshop on machine learning for information extraction* (1999), pp. 31–36. Orlando, Florida. (p. 16)

GALLEY, M. 2006. A skip-chain conditional random field for ranking meeting utterances by importance. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing* (2006), pp. 364–372. Association for Computational Linguistics. (p. 17)

GUYON, I. AND ELISSEEFF, A. 2003. An introduction to variable and feature selection. *The Journal of Machine Learning Research 3*, 1157–1182. (p. 10)

KIM, E., HELAL, S., AND COOK, D. 2010. Human activity recognition and pattern discovery. *Pervasive Computing, IEEE 9*, 1, 48–53. (p. 17)

KIM, J., OHTA, T., TATEISI, Y., AND TSUJII, J. 2003. Genia corpusa semantically annotated corpus for bio-textmining. *Bioinformatics 19*, suppl 1, i180. (p. 5)

KUDO, T. 2007. Crf++: Yet another crf toolkit. *Version 0.5 available at http://crfpp. sourceforge. net*. (p. 18)

LAFFERTY, J., MCCALLUM, A., AND PEREIRA, F. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *MACHINE LEARNING-INTERNATIONAL WORKSHOP THEN CONFERENCE-* (2001), pp. 282–289. Citeseer. (p. 17)

LING, X. AND WELD, D. 2010. Temporal information extraction. In *Proceedings of the Twenty Fifth National Conference on Artificial Intelligence* (2010). (p. 13)

MANNING, C., RAGHAVAN, P., AND SCHÜTZE, H. 2008. Introduction to information retrieval. (p. 14)

MCCALLUM, A., FREITAG, D., AND PEREIRA, F. 2000. Maximum entropy markov models for information extraction and segmentation. In *Proceedings of the Seventeenth International Conference on Machine Learning* (2000), pp. 591–598. Citeseer. (p. 16)

MITCHELL, T. AND CARBONELL, J. 1986. *Machine learning: A guide to current research*. Springer. (p. 14)

NIGAM, K. AND GHANI, R. 2000. Understanding the behavior of co-training. In *Proceedings of KDD-2000 workshop on text mining* (2000), pp. 15–17. Citeseer. (pp. 18, 30)

PUSTEJOVSKY, J., CASTANO, J., INGRIA, R., SAURÍ, R., GAIZAUSKAS, R., SETZER, A., KATZ, G., AND RADEV, D. 2003. Timeml: Robust specification of event and temporal expressions in text. In *IWCS-5 Fifth International Workshop on Computational Semantics* (2003). (pp. 8, 9, 21)

RABINER, L. 1989. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE 77*, 2, 257–286. (p. 15)

RABINER, L. AND JUANG, B. 1986. An introduction to hidden markov models. *ASSP Magazine, IEEE 3*, 1, 4–16. (p. 15)

SARKAR, A. 2001. Applying co-training methods to statistical parsing. In *Proceedings of the second meeting of the North American Chapter of the Association for Computational Linguistics on Language technologies* (2001), pp. 1–8. Association for Computational Linguistics. (p. 18)

SAURÍ, R., KNIPPEN, R., VERHAGEN, M., AND PUSTEJOVSKY, J. 2005. Evita: a robust event recognizer for qa systems. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing* (2005), pp. 700–707. Association for Computational Linguistics. (p. 13)

SAURÍ, R., LITTMAN, J., GAIZAUSKAS, R., SETZER, A., AND PUSTEJOVSKY, J. 2006. Timeml annotation guidelines, version 1.2. 1. (p. 8)

STEFANI, A. AND STRAPPARAVA, C. 1999. Exploiting nlp techniques to build user model for web sites: the use of wordnet in siteif project. In *Proc. 2nd Workshop on Adaptive Systems and User Modeling on the WWW* (1999). (p. 5)

SUTTON, C. AND MCCALLUM, A. 2006. An introduction to conditional random fields for relational learning. *Introduction to statistical relational learning*, 95–130. (p. 17)

SUTTON, C., MCCALLUM, A., AND SCIENCE., M. U. A. D. O. C. 2004. *Collective segmentation and labeling of distant entities in information extraction*. Citeseer. (p. 17)

VERHAGEN, M., MANI, I., SAURI, R., KNIPPEN, R., JANG, S., LITTMAN, J., RUMSHISKY, A., PHILLIPS, J., AND PUSTEJOVSKY, J. 2005. Automating temporal annotation with tarsqi. In *Proceedings of the ACL 2005 on Interactive poster and demonstration sessions* (2005), pp. 81–84. Association for Computational Linguistics. (p. 13)

VERHAGEN, M. AND PUSTEJOVSKY, J. 2008. Temporal processing with the tarsqi toolkit. In *22nd International Conference on on Computational Linguistics: Demonstration Papers* (2008), pp. 189–192. Association for Computational Linguistics. (p. 13)

XUAN-HIEU, P. 2005. Co-training of conditional random fields for segmenting sequence data. (p. 18)

YANAGAWA, A., CHANG, S., KENNEDY, L., AND HSU, W. 2007. Columbia universitys baseline detectors for 374 lscom semantic visual concepts. *Columbia University ADVENT technical report*. (p. 15)