Real Time Extraction of Quantitative Information from Economic news

Utkarsh Upadhyay
(Y5488)
Mentor: Prof. R.M.K. Sinha
Dept. of Electrical Engineering,
Indian Institute of Technology,
Kanpur - 208016.

April 13, 2009

Acknowledgements

The project, in hind sight, was a mammoth task.
There were various points in the execution at which it seemed that all hope was lost.
However, it was still completed in a way.
There is still much area left unexplored, and much uncharted waters to tread, but the boat has been set to sail.
There is still left a desire to have done something differently, or the wish that something had not been done the way it is done; the eternal desire to attain perfection given a *little* more time is perhaps the most human of all desires.
Nonetheless, for the state the project is in, and for their time and effort spent in reviewing the strategies and giving me new directions with respect to the problem, I would like to heartily thank Prof. R.M.K. Sinha and Prof. Harish Karnick.

Also, I would like to express my obligation to my friend, Mr. Ramnik Arora for his immense help and being a Gandalf figure. However, I am afraid I will be unable to mention everyone who has helped my here, but I would like to express my sincerest thanks to all who have been a part of this project.

Disclaimer

Presentation of news material from sources is avoided here. However, an exception is made for the Malaga news article, the excerpts of which are presented in for of a result. This is to certify that the news article belongs to MarketWatch[1] and is presented here for a non-commercial purpose according to their TOS:

> `...you may occasionally distribute a copy of an article, or a portion of an article, from a Service in non-electronic form to a few individuals without charge, provided you include all copyright and other proprietary rights notices in the same form in which the notices appear in the Services. ...`

Their entire TOS can be read at http://www.marketwatch.com/support/disclaimer.asp

[1] http://www.marketwatch.com

List of Figures

Acronyms

- *NLP* : Natural Language Processing
- *SRL* : Semantic Role Labelling
- *RSS* : Really Simple Syndicate
- *NER* : Named Entity Recognizer
- *TOS* : Terms of Services

List of Tables

Contents

1 Introduction & motivation

Extraction of data from natural language text is a very standard problem, be it extraction of links from a web page to extraction of topic from a body of text. A similar attempt is made here to attempt to extract *quantitative information* from Economic news, which is released in real time over the Internet. The importance of this is undoubted, and similarly undoubted are the state-of-the-art techniques in existence. This task of extracting information could have been done on any kind of news/natural language text but *economic news* has a special significance.

1.1 Interactive Broker's Algorithmic Trading Olympiad

Every year, since 2002, **Interactive Broker** has been running an on-line competition of algorithmic trading over equity markets. It has run for a about three months, i.e., January, February and March of each year since then. Ramnik Arora (M. Sc. MTH) and I took part in the competition last year and wish to do it again this year. According to the guidelines of the competition, one is allowed use of market news in making our decisions for trading, as long as the decisions are taken by help of a program. I intend to make the program usable as a helper module till then.

2 The task

As we had attempted trading using only numeric information in the last year, we had met with only limited success. It made us clearly realise the importance of Quantitative information (*next Quarter estimates, profits, etc.*) being released in real time over the Internet. We can attempt to perform *Sentiment analysis* over the news, but such information though helpful, is generally inaccurate. *Ankit Soni*[2][1] did his thesis on the topic, and his results were that after extensive testing, he was able to attain about 56% accuracy. His classifier identified each news as positive, negative, or as neutral.

However, if we are able to extract quantitative, instead of qualitative, information from the news, then by comparing it with the previous estimates, we obtain numeric gauge which is easier to work with and they generally have a more predictable impact on the market prices of the equity. Cultivating information from more than one source would benefit us since if the number of sources is large, then it would no longer remain possible for a human to out-perform the machine in terms of content handled. Also, it would be interesting to note the different news from different sources and question their reliability.

2.1 Existing work

The academic work in the area is limited to financial ontology[2], and is generally done behind closed doors. Among the better known open implementations is ***FASTUS* (1993)**[3]. It was capable of filling in database entries based on the Natural Language news it processed. However, it did not use much of the syntactic information latent in natural language. We here attempt to provide nearly the same application utilizing the syntactical information for better results as well as an academic exercise.

After FASTUS, most of the work in the field was closed source, but with CoNLL[4], and semantic role labelling, interest in the task has been regenerated. The problem being addressed at CoNLL, however, is in a very general context where the *frame* structure of all the verbs is being explored. Hence, in this project, a subset of SRL is being attempted, restricted to only those verbs which are common in Financial news articles and with frames which only relate to quantitative information.

[2]Working under Prof. Harish Karnick, 2005

2.2 Problem statement

Extract the quantitative figures of the primarily the following:

1. Revenues
2. Profits/Losses
3. Turnover
4. Earnings per share

from natural-language news released over the Internet. The extracted news should also provide a context for the information including the entity name, the time frame of the news, and the *kind* of quantitative information (revenue, profit, turnover, etc.). Also, to perform this task in real-time, as the markets exhibit extremely quick assimilation of any news.

3 Proposed solution

The following strategy was proposed for solving the set problem.

3.1 First Stage: Simplification

We first break the natural language news to a set of independent sentences. The task continues to further subdivide parts of the sentences to arrive at as succinct phrase as possibly can fill in a template. This is done to limit our search space (and populate the feature space, as we will see). For example:

There is significant political turmoil in US and the price of oil has been quavering after the recent wars, and, hence, the estimates for Q3 dropped by 17%.

to

1. *There is significant political turmoil in US*
2. *the price of oil has been quavering after the recent wars*
3. *hence, the estimates for Q3 dropped by 17 % .*
4. ***Garbage:*** *, and , .*
5. ***Garbage:*** *and*

-Yahoo news article.

The breaking can be understood by looking at the following tree diagram of the sentence (figure 1), and cleaving the tree at all intermittent 'S' nodes (green in color).

3.1.1 Non-natural language news

However, it often happens that the news released itself contains quoted tables from the company's reports, etc. which are not parsable. Hence, they need to be treated differently and this stage itself.

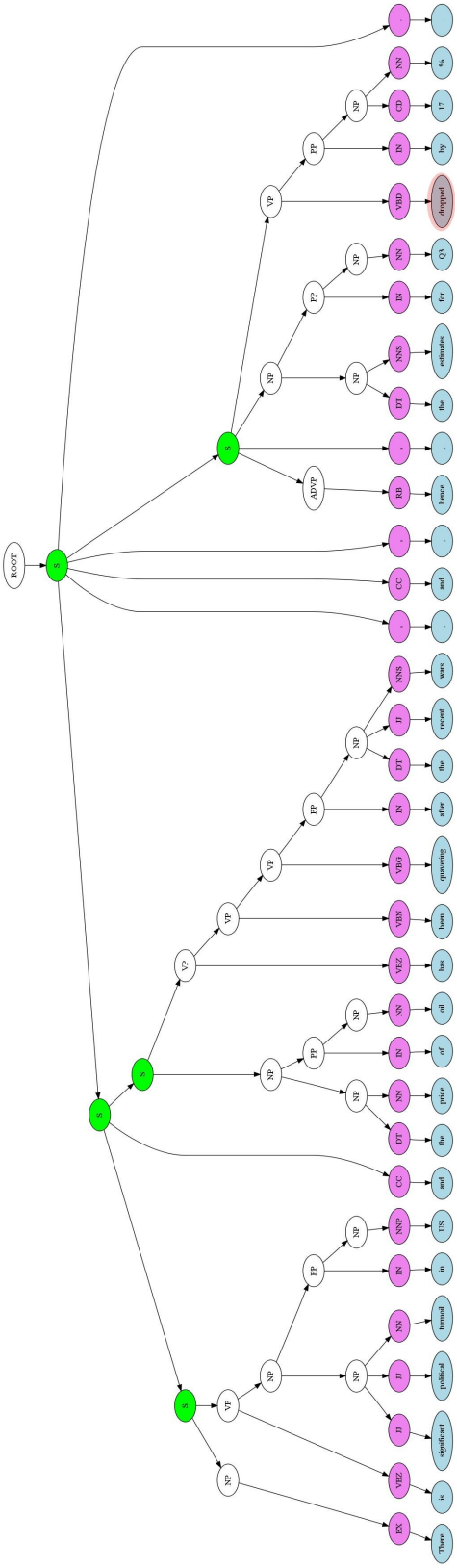

Figure 1: A parse Tree

3.2 Second Stage: Feature extraction

It is generally agreed in academic circles that the verb is the most significant part of a sentence, which determines what *role* the other parts of the sentence play and what is the semantic meaning of the sentence.

In the parse of the given sentence (figure 1), we can see that the *structure* of information *around* the verb:

- "for Q3" *(The time)*: Verb (VBD) ↑ Verb Phrase (VP) ↑ Sentence (S) ↓ Noun Phrase ↓ Propositional Phrase (PP)
- "by 17%" *(The quantity)*: Verb (VBD) ↑ Verb Phrase (VP) ↓ Propositional Phrase (PP)
- ...

This *tree path* will be the primary feature which would be used for the task and its effectiveness would be tested before attempting to use other features/techniques.

The motivation behind choosing this feature was that it is considered one of the most important features in SRL[5], which is very similar to our task, as could be seen in figure 2.

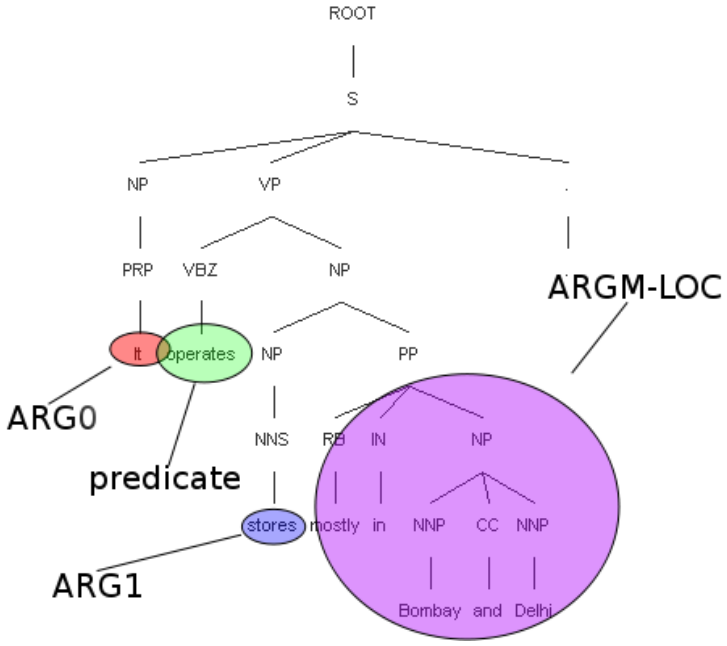

Figure 2: An example of SRL

To perform the task, hence, our requirements (and the components that play the respective parts) are shown in table 1.

However, among other tasks, manual tagging of a multitude of sentences will need to be done before one can attempt to perform the task automatically.

3.3 Third Stage: Template Design

Tagging will be done at two stages corresponding to the two stage architecture of the project.

Task	Component Used
Generating the parse tree	Stanford Parser
News Updates	Self made RSS reader
News Extractor	Self made filters using HTMLParserLib
Interface/GUI	Self made on Java
Supplementary requirements:	
Storage format	Self designed tagging format
Tagger	Self designed GUI

Table 1: Requirements for the project

Sentence Level

Possible tags are:

- No tags (Useless)
- Tagged as Useful
- Tagged as deceptive

These tags should help one in determining which sentences to pick and which to disregard, helping in fine tuning the summarizer. Also, with the GUI's help, the tagging boundaries are made to coincide with proper syntactic boundaries, i.e., the parts selected will be complete sub-trees of the original parse tree.

Tags

All sentences should be able to fill the following template using the information contained therein and in neighbouring sentences:

- Predicate
- Arg-TMP
- Arg1 (absolute_value)
- Arg2 (relative_value)
- Arg3 (direction)

However, at the sub-sentence level, the following tags are also being assigned:

- Keyword : These are the non-verb predicates. Predicates automatically qualify for keywords.
- Named entities

to help us in making a database of the corresponding. Also, the tags within themselves have a few sub-tags.

4 Time line

The work on the project had begun in the right earnest from August itself, when a survey of sorts of what was available in the market was conducted, and the best and/or most convenient tools were picked for the task. A succinct summary of the tasks accomplished in various time periods is given below:

- **August - September '08**: Familiarising with the Stanford NLP tools and formulating the problem statement.
- **September - October '09**: Made a HTML parser using *HTMLparserlib* for *finance.yahoo.com* articles[3] and a simple *RSS* parser. Also, was able to fetch news from the Internet and save it (and process it) in real time.
- **October - November '08**: Using the Stanford Parser, making a Summarizer tool for helping with *Tregex* expressions and fine tuning the application for real time performance. Helped in finalizing the strategy to be adopted.
- 23^{rd} **Nov -** 8^{th} **Jan. '09**: Worked on tagging news. Designed the format of tags files (an altered XML format). As the Stanford tagger did not ensure preservation of syntactic units while annotating text, made a GUI to assist in correct tagging. Developed an application to inspect the difference between Factored parse trees and PCFG parse trees using the Stanford Parser. Had multiple false starts, had to back-trace and fix the tagging many times. Also worked on another trading strategy[4] with Ramnik Arora which was to be deployed for online trading. Attained the direct connection to the Internet and was able to run the *Interactive Brokers* application on it.
- 8^{th} **Jan -** 20^{th} **Jan. '09**: Worked on integration of Ptolemy[5] (a Java based general purpose simulator, whose Discreet Event model simulated our trading platform) and the IB Application. Made a significant contribution to the Open Source software from UC Berkley.
- 20^{th} **Jan -** 9^{th} **Feb. '09**: Worked on the *Sentence Annotator*, which integrated the tagged information and the tree structure of parsed news sentences. The data was found to be surprisingly sparse. Made a GUI to guide us through the information ourselves and assist in verification. As the tagging data file did not conform to the standard XML format and as the order of elements was important here, wrote a small XML like parser for the tagged files.
- 9^{th} **Feb. -** 20^{th} **Feb. '09**: Wrote an extensible annotation assimilator, which could save the relationship between the tree structure *(TreePaths)* and the *tags*. Wrote a *PathFollower* which was able to look for similar structures in new trees. Also, integrated the two parts of the project: the online RSS, news parser and the automatic annotator together for real time testing.
- 20^{th} **Feb. -** 8^{th} **Mar. '09**: Deployed the system and removed various bugs. Executed a few trades based on the news outcomes.

5 Groundwork

Reading about the various tools available online was started in August when various platforms and tools were considered for performing the task. Various interfaces were looked at, including the re-ranking parser of Charniak[6] written in C/C++, the Natural Language Toolbox written in Python, etc. However, one of the constrains was that our trading system had to be Ptolemy based[7], which itself was written in Java. Hence, the Stanford Parser[8], also written in Java was an obvious choice. It also could be easily integrated with other tools available from the same Group, like the Named Entity

[3] This format was changed on 25^{th} March '09, thereby breaking the news fetcher.
[4] http://home.iitk.ac.in/ ramnik/IIMA-Proposal.pdf
[5] http://home.iitk.ac.in/ ramnik/gsoc.html

Feature	Value
Parser	PCFG
Sentence length	80
Output format	Penn (without scores)

Table 2: Parameters chosen for the Parser

Recognizer[9]. An implementation of Tregex, and Tsurgeon had also been done and was present as a part of the Stanford package itself, while the other packages depended on external tools for the same. HTMLParserLib[10], an open source project meant for performing tasks on HTML documents, is also Java compatible.

Apart from these advantages, as both Ramnik and I were very comfortable with Java as a programming language, we chose it for the task.

6 Pattern searcher and news fetching

After the ground work was laid, came the phase of familiarising myself with the tools. The first step taken in the direction was making a HTML filter which could extract the textual news from the HTML pages fetched. This was made using the HTMLParserLib for both, Yahoo News[6] and MarketWatch News[7]. The MarketWatch parser still works as of April 13, 2009, but the compatibility with the Yahoo parser was broken when Yahoo decided to change its web page format, which is likely to change again.

Also a generic RSS reader was designed to read the RSS feeds of both Yahoo News as well as MarketWatch. The work later, however, continued with Yahoo only.

7 Summarizer

After the exercise of downloading new news articles as they appear on the RSS feed and then extracting the text information from it was over, a Pattern Searcher was made which could search for a Tregex pattern on the sentences present in a news article. The news articles first had to be parsed to a tree structure, again done using the Stanford Parser. It was made with a dual purpose, firstly it familiarised us with the API of the Stanford Stanford Parser and secondly, it acted as a test-bed for various Tregex expressions that were used later. Furthermore, it was also seminal in understanding what settings for the parser would be the best for us. After much experimentation, the settings settled on were those present in table 2.

It could also have been used as a sentence summarizer which could separate out sentences with a certain structure (or which contained Quantitative information based on the presence of *QP* and/or *CD* node). This work was completed by the end of the month of November '08.

8 Manual tagging

While attempting to tag the text according as per the rules of section 3.3, much difficulty was faced. Firstly, Stanford annotator[11] could not be used because it does not ensure that the tagging done will have proper syntactic structure, i.e., the tags may not contain exactly a subtree of the parse tree. To ensure the same, another tagger (see figure 3) had to be made.

[6] http://finance.yahoo.com
[7] www.marketwatch.com

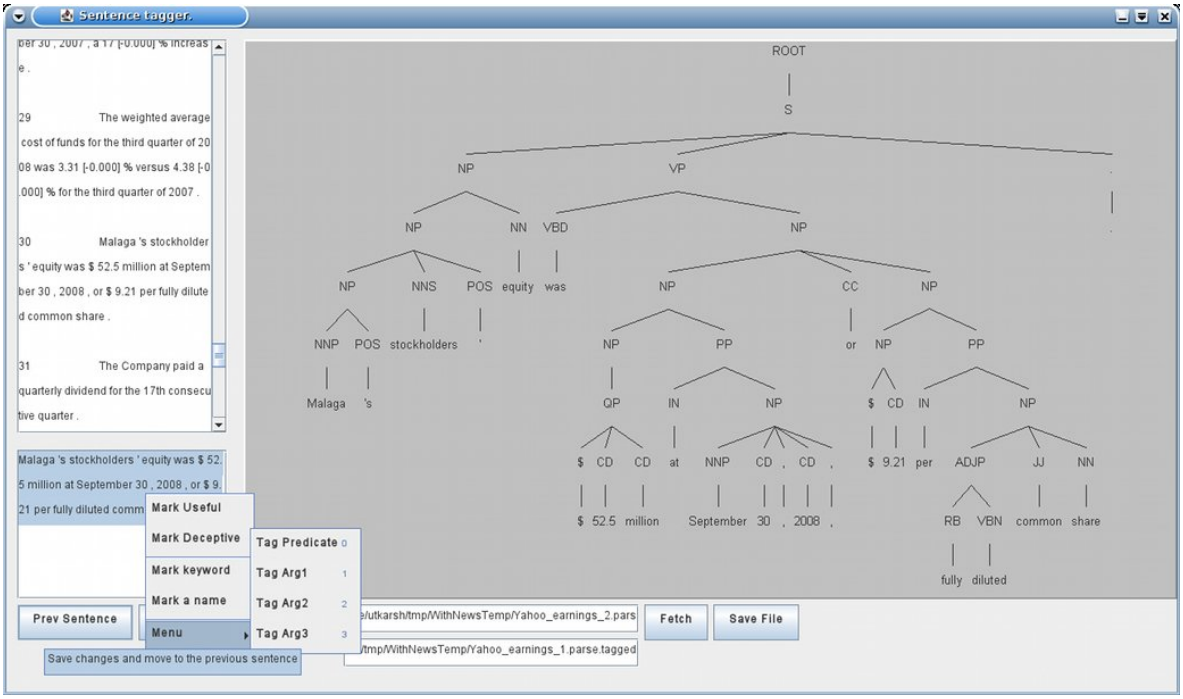

Figure 3: GUI for the Manual Tagger

It can be seen in figure 3 that the news (in the top left text area) was parsed sentence by sentence. While working on one sentence, its parse tree was displayed on the JFrame on the left, and the text of the sentence was displayed in the lower left text area. To tag a part of the sentence, one had to select that part and then select the desired tag which would be all present in a pop-up menu. However, it was still the duty of the person performing the tagging to ensure that the tagged part of the text was consistent in its syntactical structure. The Tree display has more space allotted to it than the other text boxes because in case of complicated sentences, the parse tree may become fairly complex. The tagged news articles were saved in a separate file, and the tagging was done in a XML format. A typical sentence after tagging would look like:

```
<useful>
    <keyword>Shares</keyword>
    <direction>
        <predicate>fell </predicate>
    </direction>
    <relative_amount>
        <absolute_amount>$ 3.91</absolute_amount>
        , or
        <percent>9.5 percent</percent> ,
    </relative_amount>
</useful>
to close at
    <absolute_amount>$ 37.46 .</absolute_amount>
```

The GUI certainly made syntactically correct tagging possible, but it was still a tiresome job and even after weeks of effort, we were able to collect together and tag only about 100 sentences from numerous news articles which could be useful for us.

Initially the focus was on extracting only the future or the correct news and, hence, sentences which contained quantitative information, but of the past, were labelled as *deceptive*. However, owing of the sparsity of the data, the *deceptive* and *useful* tags were later discarded.

Also, after a significant part of tagging was complete, it was seen that the factored parses led to markedly different results than the PCFG parser. Another GUI was made for observing the differences, one of the most remarkable being the example shown in figure 4. The left sub-tree is the PCFG parse, while on the right is the Factored Parse. The PCFG parse is clearly of an inferior quality here.

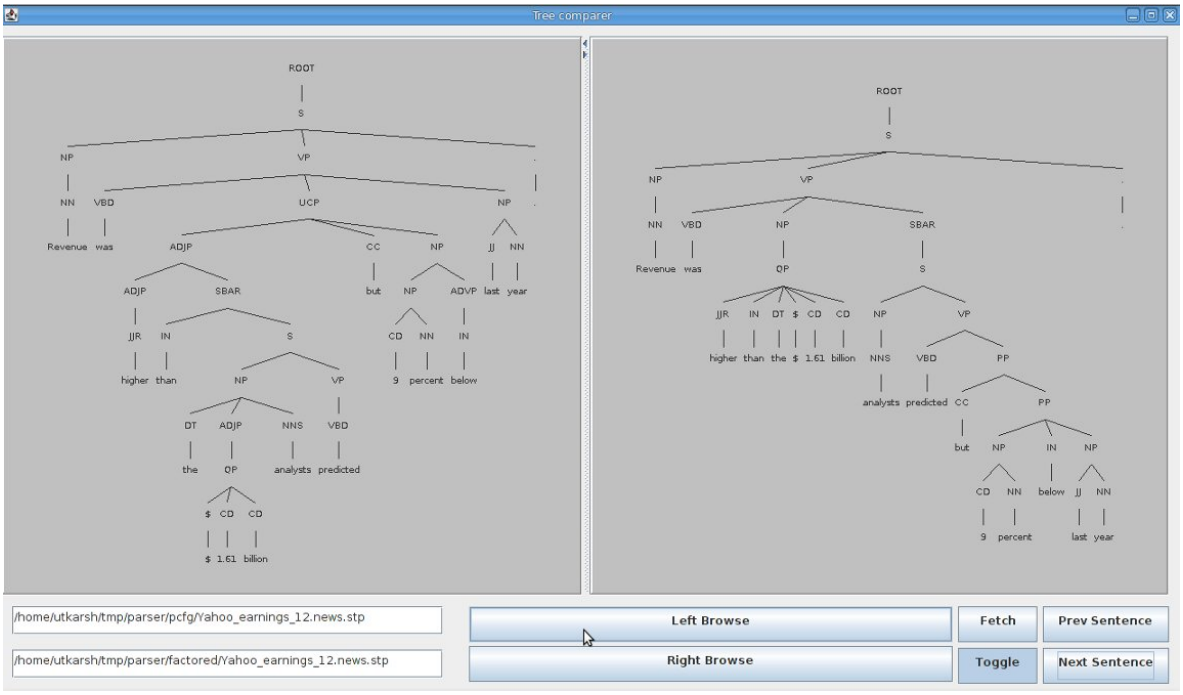

Figure 4: Difference between the PCFG and the Factored parse

However, as the Factored parsing took too long, it was decided that the PCFG parsing would be retained.

8.1 Separate trading strategy

Along with the tagging work, we (Ramnik and I) were also working on a separate trading strategy which tried to emulate ETF arbitrage techniques to sectoral trading. A neural network was used to predict the current fair price of a target security using stocks in its sector and any deviation of the true price from the computed fair price was traded upon. The idea was later formalized and had been submitted to NSE, India for research funding and also has been accepted for presentation in the Advanced Data Analysis, Business Analytics and Intelligence[8].

9 Preparing for the competition

In parallel with the tagging strategy, preparations for the competition were also on. The competition was scheduled to start from 8^{th} Jan. '09, and we could obtain a direct connection to the Internet only

[8]http://www.iimahd.ernet.in/icadabai2009/

on the same day. It was necessary to obtain the connection for running the application because it was not designed to work from behind a proxy. What the IB platform provided us was a port over which we could transfer information to the application running on our system and then the application executed the requested actions by sending the proper requests to the main IB server[9].

9.1 Integration with Ptolemy

Thereafter, next two weeks were spent in setting up Ptolemy and integrating it with the IB's server application. Ptolemy is a general purpose event simulator currently under development at UCB. As most of our trading strategies were based on a new price information about any stock, we could easily simulate it as an discreet event system, with the incoming *price ticks* triggering events which lead to calculations and then, finally, a trade decision.

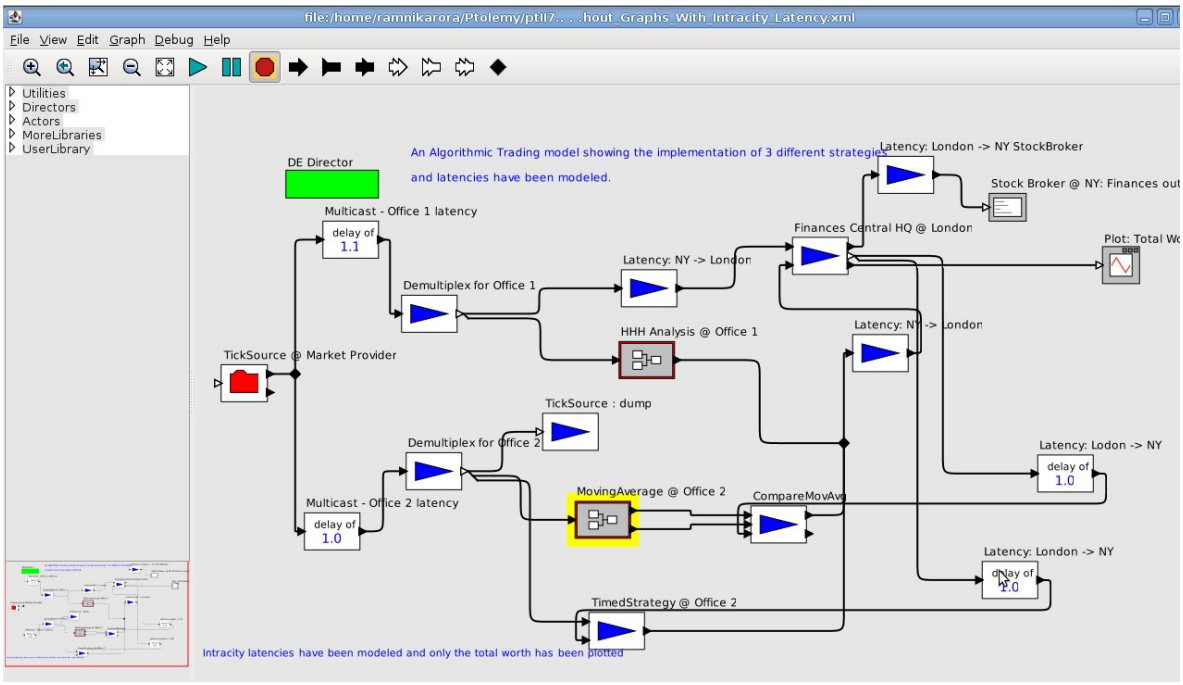

Figure 5: Ptolemy

In the process, we also made significant contributions to the open source project.[10].

10 Sentence Annotation

After the infrastructure for the competition ready was ready, work continued on the sentence annotator which could read in a parse tree, its tagged version and then *understand* which sub-tree was actually the one which was tagged. Hence, before one could combine the tagging information and the structural information about a sentence, one had to *understand* the tagging information, for which, on a glance, a normal XML reader would suffice. However, that was not the case.

[9] gw4.ibllc.com
[10] http://home.iitk.ac.in/ ramnik/gsoc.html

10.1 XML Like Parsing

There are, strictly speaking, two kinds of XML parsers:

- **DOM parsers**: They parse the entire document to a tree structure with the tags at nodes, but lose the order of the tags.
- **SAX parsers**: They too process an entire document together, and to work with them, one would need to make data structures which would hold the entire parse information, managing which would be as tough as writing the parser itself.

However, it was later discovered that determining the *correct* sub-tree of the parse tree of the sentences, would need more information than merely the words contained in the tag: it would require the preceding as well as the succeeding word, and would need to know that there are none, if so is the case.

Hence, because the requirements were very limited as well as specific, a XML like parser was designed. However, it happened to be a tougher task than first imagined. After a series of edits, and JUnit tests, the following pattern for extracting the preceding word, the succeeding word, and the words inside the tag as groups [1,3,5]:

```
Pattern pat = Pattern.compile(
    "(\\b?[^\\s>]+)?[\\s]*([\\s]*<[^<]*?>)*[\\s]*<"   // Group 1 and 2
    + tag +
    ">[\\s]*(.*?)[\\s]*</"                            // Group 3
    + tag +
    ">[\\s]*(<[^>]*?>[\\s]*)*[\\s]*([^\\s<]+\\b?)?"); // Group 4 and 5
```

Out of many reasons why it was difficult to arrive at was that the certain tags may not have a succeeding or preceding word and they might be enclosed inside other tags (e.g. <direction><predicate> **fell** </predicate></direction>). After the tagging was complete, the following verbs (*predicates*) had occurred in the tagged text (unstemmed):

```
rose
fell
was
tumbled
saw
dropped
reported
expect
posted
cut
expect
closed
```

10.2 Annotation

Soon after task of extraction of tags along with their preceding/succeeding was over, a Tregex pattern was designed to search for the sub-tree inside a given sentence. The following Tregex expressions were employed for the task, in that order:

```
( __ << ( word1 . word2 . word3 . word4 ))
```

Pattern	Meaning
__	matches any node
A << B	A dominates B
A . B	A precedes B
A , B	A succeeds B

Table 3: Description of the Tregex expression

```
(( __ << ( word1 . word2 . word3 . word4 )) , preceding_word)
(( __ << ( word1 . word2 . word3 . word4 )) . succeeding_word)
((( __ << ( wrd1 . wrd2 . wrd3 ) , preceding) . succeeding)
```

where the explanation of the Tregex is given in table 3 and:

1. *word1, word2, word3, word4 ...* are words present inside the tag.
2. *preceding_word* is the word that comes before the words inside the tag (may be *null*).
3. *succeeding_word* is the word that comes after the words inside the tag (may be *null*).

The tree that matched the last of these expression was chosen as the actual sub-tree. This strategy had to be employed, because while tagging, the sentences had not been broken into the simplest form (as described in Section 3.1). Hence, after simplification of sentences, the preceding/succeeding word may no longer be a part of the sentence. The use of the expression is made clear by the following:

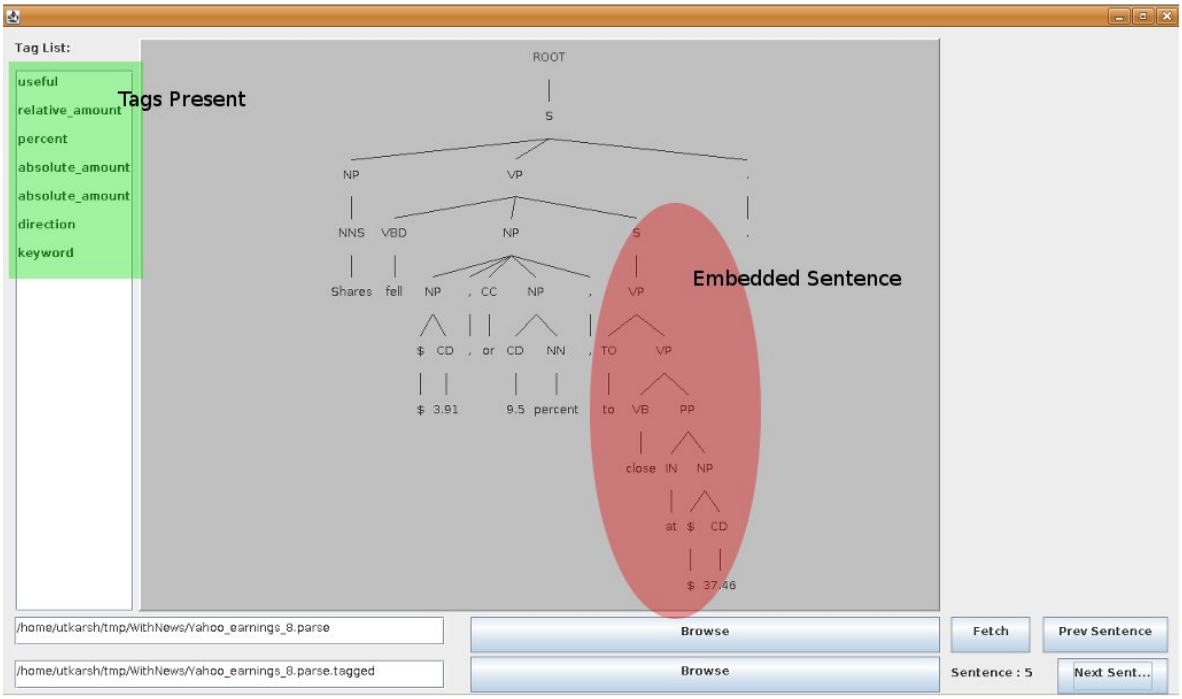

Figure 6: Tags and embedded sentences

After the sub-tree was found, the path on the parse tree that had to be followed to arrive at the thus found *tagged* node was determined using Breath first search.

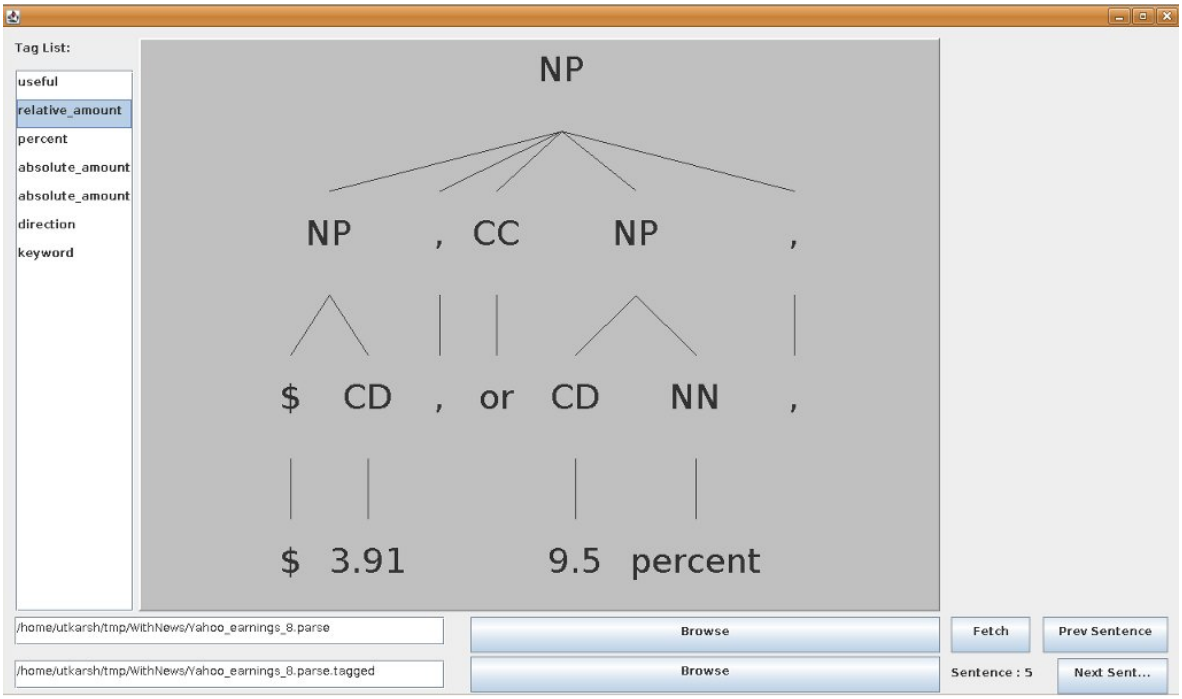

Figure 7: Example of a relative amount in sentence of figure 6

The Parser API is a little finicky here. It only returns paths of *'objects'* which are subtrees of the original tree, not the *'subtrees'* by content: A typical issue of using == equivalence instead of *.equals()*. It understandably saves some computation on regular searches. However, owing to it, the code written has some overhead.

It can be seen in Figure 10 that in the actual tree (figure 9), one needs to start from the *VBD* node, then ascend to *VP* and then descend to *NP* which is the relative amount.

Finally the results were stored in a file in the following format:

```
For the relation:be -> absolute_amount
      VBD(up)VP(down)NP(down)NP : 2
      VBD(up)VP(down)PP(down)NP(down)PP(down)NP : 1
```

10.2.1 Sparsity of data

After the due analysis, it was seen that the thus tagged data was very sparse, and there was only one instance of a verb (*predicate*, which was '*be*') and a tag (*absolute_amount*) with a certain structure occurring more than once. Hence, attempting to learn the characteristics of the path did not seem the ideal path to follow. Hence, the strategy was modified slightly to deal with the new circumstances and time frame.

11 Modified strategy

The next obvious extension to the system was making a *PathFollower* which would be capable of following the paths found above in new trees and come up with options. It was implemented and worked thus:

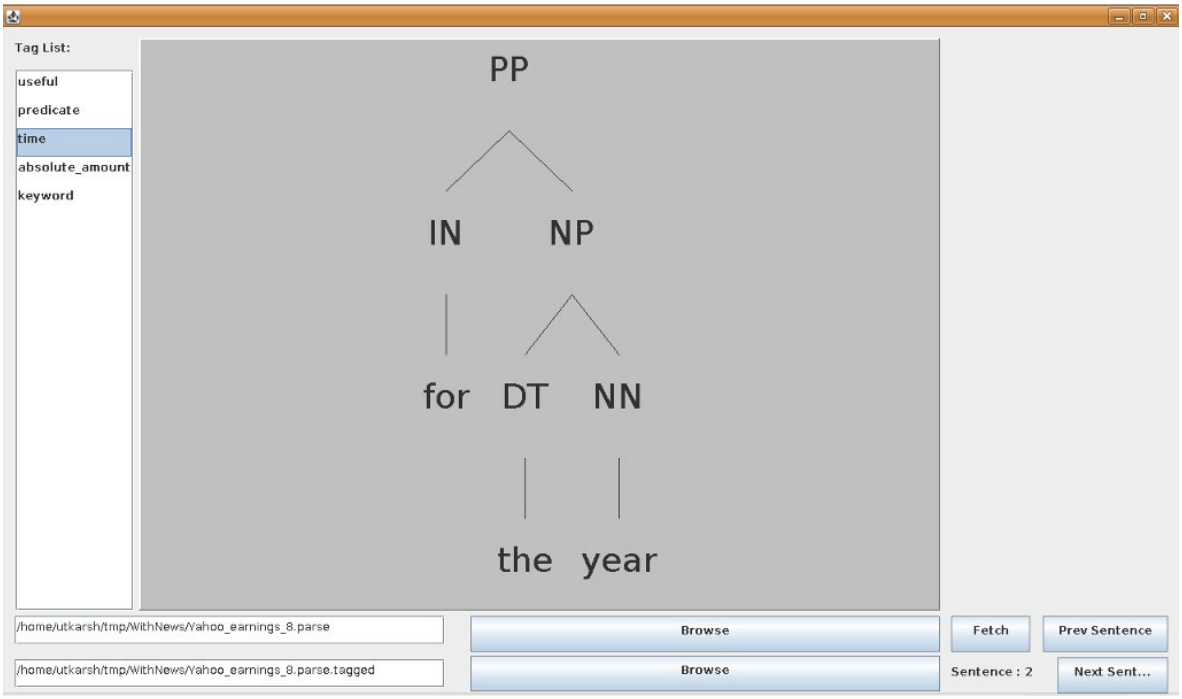

Figure 8: Example of time

```
In File : 2 for predicate : be and
path/tag: VBD(up)VP(down)NP(down)NP/absolute_amount
there are 3 possibilities:
        $ 1,783,000
        an increase of $ 303,000
        20 % from net income of $ 1,480,000
```

The file in which this search was made was *2*, the predicate was *be* and the element being searched for was *absolute_amount*. The path predicted for it was: *VBD(up)VP(down)NP(down)NP* and in the statement, three possible fits to this tree path were found, which are listed.

The original sentence was:

```
(ROOT
  (S
    (S
      (NP (NNS PALOS))
      (VP (VBZ VERDES)
        (NP
          (NP
            (NP (NNS ESTATES))
            (, ,)
            (NP (NNP Calif.))
            (: --))
          (PRN (-LRB- -LRB-)
            (NP (NNP BUSINESS) (NNP WIRE))
            (-RRB- -RRB-)))))
```

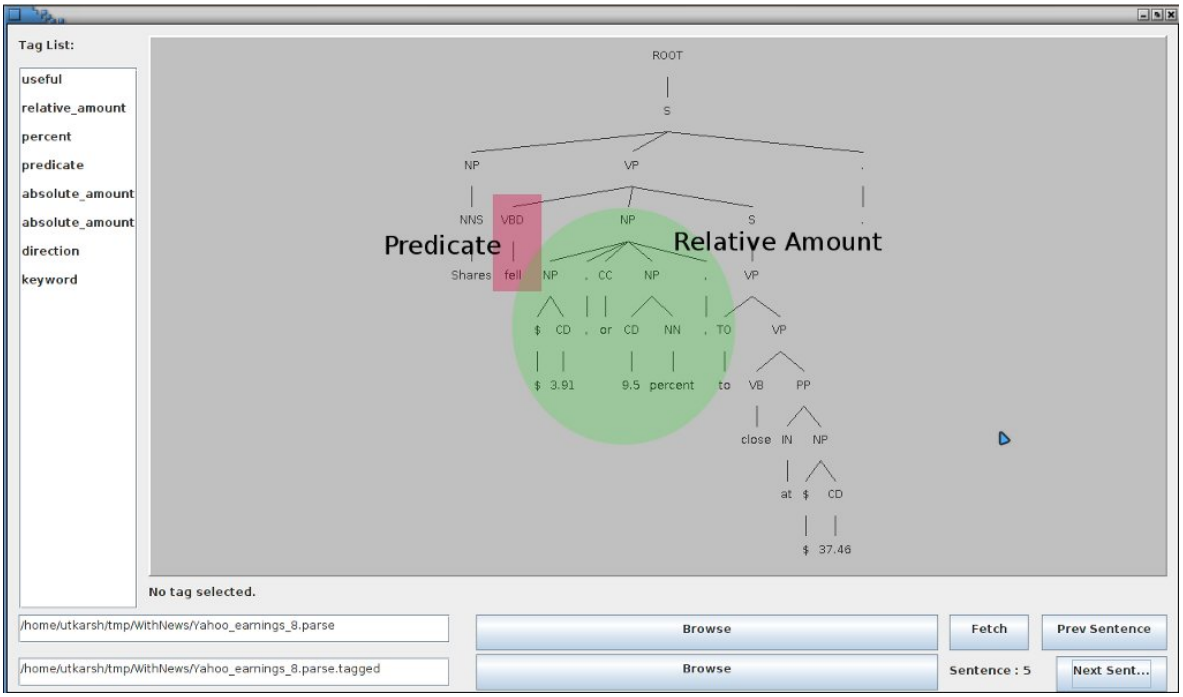

Figure 9: The whole sentence

```
(: --)
(S
  ...
  (NP-TMP (NN today))
  (VP (VBD reported)
    (SBAR (IN that)
      (S
        ...
        ))
(. .)))
```

or in words:

> PALOS VERDES ESTATES , Calif. – -LRB- BUSINESS WIRE -RRB- – Malaga Financial Corporation -LRB- OTCBB : MLGF - News -RRB- , the parent company of Malaga Bank FSB , today reported that net income for the quarter ended September 30 , 2008 was $ 1,783,000 , an increase of $ 303,000 or 20 % from net income of $ 1,480,000 for the quarter ended September 30 , 2007 .

The information thus extracted is not perfect, but is close enough for a human reader to comprehend. Hence, the strategy did not remain completely automated beyond this point. Owing to sheer sparsity of data, it was no longer possible to allow the program to attempt to fill the complete template by itself. Hence, as an appendage to the existing infrastructure, a *Dictionary searcher*, for keywords and names, and a PatternSearcher for time references, were made to aid in completing the template. They, however, do not make use of the structural properties of the sentence.

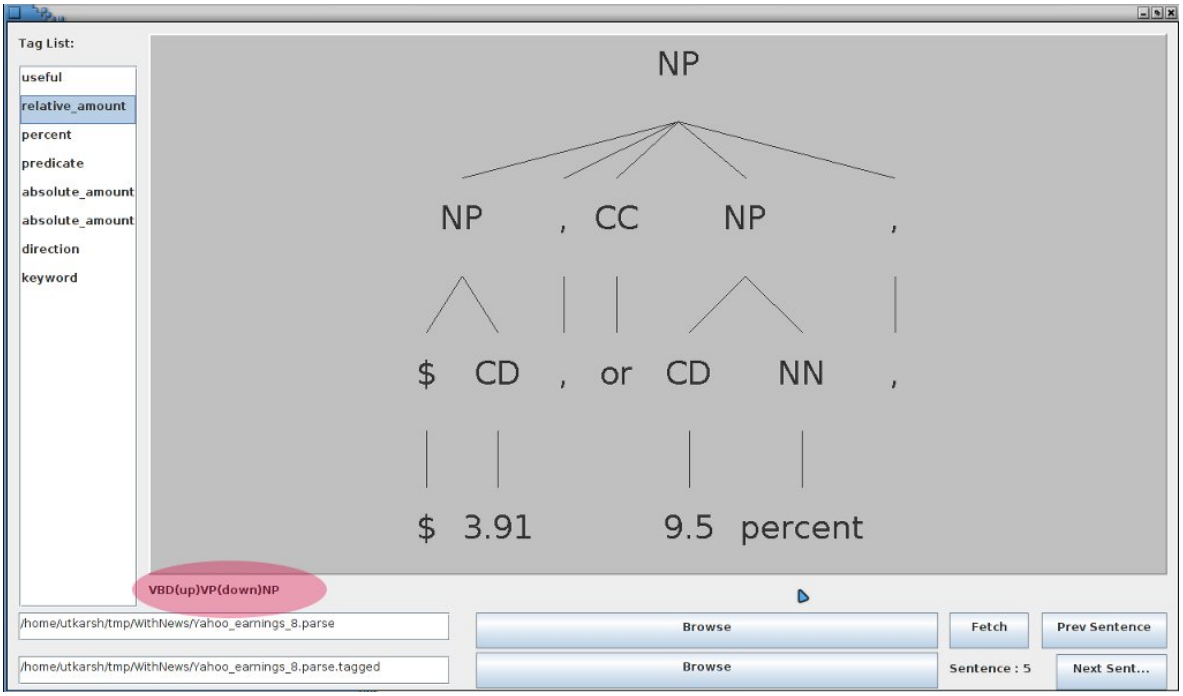

Figure 10: A path from the *predicate* to the tag (*rel amt*)

11.1 OnlineNewsReader

Finally the two parts of the project were amalgamated: the online news fetcher/text extractor and the PathFollower (See figure 11)

The buttons on the Panel in figure 11 are explained below:

- *Process News* : Scans the news for tags. The pop-up box cycles through all the tags found in the news articles sentence by sentence.
- *Refresh* : Fetches any new news articles using the Yahoo RSS feed.
- *Next / Prev* : Cycles through the news articles that have been fetched.

12 Deployment

The system went online on 20^{th} Feb. '09. Periodically scans on the RSS feed were made manually and the fetched news articles were scanned for presence of any tags. If a structure was found in the article, then based on the context of the sentence or by comparing the values to the previous estimates, trades were committed and the position was closed at the end of the day, irrespective of profit/loss. Even though it is widely accepted that the market shows extremely quick assimilation of news, our trades yielded profit with consistency.

We committed 18 trades based on the news filtered using out system over the two weeks extending from 23^{rd} Feb. to 6^{th} Mar. '09, and 16 of these resulted in a profit. As a sample, the following news article is about Malaga Financial Corp. which resulted in a profit. The system found the following tags in the news; the first part tells which tags were found using the tree structure, then the sentence follows and, finally, any names/time references found by dictionary searching:

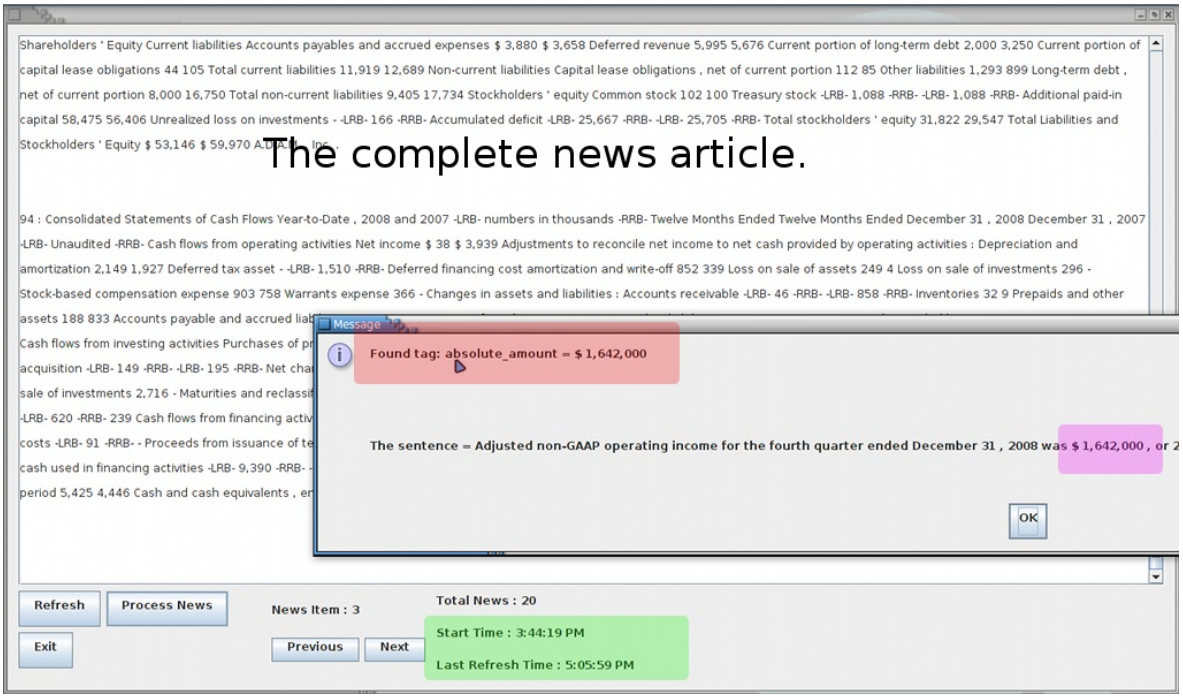

Figure 11: Online News Reader

1. • **Found tag**: predicate = was
   • **Found tag**: absolute_amount = $ 1,783,000
   • **Found tag**: absolute_amount = an increase of $ 303,000
   • **Found tag**: absolute_amount = 20 % from net income of $ 1,480,000
   • **Found tag**: keyword = net income for the quarter ended March 31 , 2009
   • **Found tag**: keyword = the quarter

   PALOS VERDES ESTATES , Calif. – -LRB- BUSINESS WIRE -RRB- – Malaga Financial Corporation -LRB- OTCBB : MLGF - News -RRB- , the parent company of Malaga Bank FSB , today reported that net income for the quarter ended March 31 , 2009 was $ 1,783,000 , an increase of $ 303,000 or 20 % from net income of $ 1,480,000 for the quarter ended September 30 , 2007 .

   • **Name Found** in sentence = BUSINESS WIRE
   • **Time Found** in sentence = March 31
   • **Keyword Found** in sentence = net income

---

2. • **Found tag**: predicate = reported

   PALOS VERDES ESTATES , Calif. – -LRB- BUSINESS WIRE -RRB- – Malaga Financial Corporation -LRB- OTCBB : MLGF - News -RRB- , the parent company of Malaga Bank FSB , today reported that net income for the quarter ended March 31 , 2008 was $ 1,783,000 , an increase of $ 303,000 or 20 % from net income of $ 1,480,000 for the quarter ended September 30 , 2007 .

   • **Name Found** in sentence = BUSINESS WIRE
   • **Time Found** in sentence = March 31
   • **Keyword Found** in sentence = net income

3.
- **Found tag**: useful = The Company 's allowance for loan losses at January 31 , 2009 was $ 2,638,000 or 0.38 % of total loans compared to $ 2,312,000 or 0.36 % of total loans as of September 30 , 2007 .
- **Found tag**: predicate = was
- **Found tag**: absolute_amount = $ 2,638,000
- **Found tag**: absolute_amount = 0.38 % of total loans
- **Found tag**: relative_amount = $ 2,638,000
- **Found tag**: relative_amount = 0.38 % of total loans
- **Found tag**: percent = 0.38 %

The Company 's allowance for loan losses at January 31 , 2009 was $ 2,638,000 or 0.38 % of total loans compared to $ 2,312,000 or 0.36 % of total loans as of September 30 , 2007 .

- **Name Found** in sentence = null
- **Time Found** in sentence = September 30
- **Keyword Found** in sentence = losses

4.
- **Found tag**: useful = The increase in operating expenses was primarily attributable to the personnel and operating costs of Malaga 's new branch in San Pedro , .
- **Found tag**: predicate = was

The increase in operating expenses was primarily attributable to the personnel and operating costs of Malaga 's new branch in San Pedro , which opened in April 2008 .

- **Name Found** in sentence = null
- **Time Found** in sentence = April 2008
- **Keyword Found** in sentence = null

5.
- **Found tag**: useful = Malaga 's total assets were $ 737 million at March 31 , 2009 compared to $ 709 million at March 31 , 2007 , an increase of $ 28 million .
- **Found tag**: predicate = were
- **Found tag**: absolute_amount = $ 737 million

Malaga 's total assets were $ 737 million at March 31 , 2009 compared to $ 709 million at March 31 , 2007 , an increase of $ 28 million .

- **Name Found** in sentence = null
- **Time Found** in sentence = March 31
- **Keyword Found** in sentence = null

6.
- **Found tag**: useful = The loan portfolio at September 30 , 2008 was $ 702 million versus $ 645 million at September 30 , 2007 , an increase of $ 57 million .
- **Found tag**: predicate = was
- **Found tag**: absolute_amount = $ 702 million
- **Found tag**: absolute_amount = an increase

The loan portfolio at September 30 , 2008 was $ 702 million versus $ 645 million at September 30 , 2007 , an increase of $ 57 million .

- **Name Found** in sentence = null
- **Time Found** in sentence = September 30
- **Keyword Found** in sentence = null

7. • **Found tag**: useful = The weighted average cost of funds for the third quarter of 2008 was 3.31 % versus 4.38 % for the third quarter of 2007 .
   • **Found tag**: predicate = was
   • **Found tag**: relative_amount = 3.31 %
   • **Found tag**: keyword = The weighted average cost of funds for the third quarter of 2008

   The weighted average cost of funds for the third quarter of 2008 was 3.31 % versus 4.38 % for the third quarter of 2007 .

   • **Name Found** in sentence = null
   • **Time Found** in sentence = third quarter
   • **Keyword Found** in sentence = null

---

8. • **Found tag**: useful = Malaga 's stockholders ' equity was $ 52.5 million at September 30 , 2008 , or $ 9.21 per fully diluted common share .
   • **Found tag**: predicate = was
   • **Found tag**: absolute_amount = $ 52.5 million at September 30 , 2008 ,
   • **Found tag**: absolute_amount = $ 9.21 per fully diluted common share

   Malaga 's stockholders ' equity was $ 52.5 million at September 30 , 2008 , or $ 9.21 per fully diluted common share .

   • **Name Found** in sentence = null
   • **Time Found** in sentence = September 30
   • **Keyword Found** in sentence = null

---

9. • **Found tag**: predicate = was
   • **Found tag**: absolute_amount = all applicable regulatory capital requirements

   As of September 30 , 2008 , Malaga Bank was in compliance with all applicable regulatory capital requirements and was deemed " well-capitalized " under applicable regulations , with a risk-based capital ratio of 13.83 % .

   • **Name Found** in sentence = null
   • **Time Found** in sentence = September 30
   • **Keyword Found** in sentence = null

There are about 40 sentences in the entire news article. Also, Malaga was not recognized as a name because it was the first time a news article was obtained for it. There are a few noticeable facts about the system latent in the output:

- In items 1,2: the sentence is the same, but the predicates are different. However, it was unable to 'find' any fit for the second predicate *'report'*.
- Named Entity recognition and time reference search is futile when there are more than one instance of any present, evident in sample 2.
- The system was able to pick out *relative_amount* and the *absolute_amount*.
- In item 3, as *relative_amount* and *absolute_amount* share the same tree structure, they are confused with each other. Clearly more features are required to differentiate between the two.
- Items 4 and 9 are false positives.

13 State of Code

The project has been built using the NetBeans IDE and follows the Object Oriented approach to solving problems. There are multiple sub-projects which are combined together finally in the project *AutoQExtractor* (Refer to figure 12). However, some of the dependencies are needed only for Unit testing and for running the system offline.

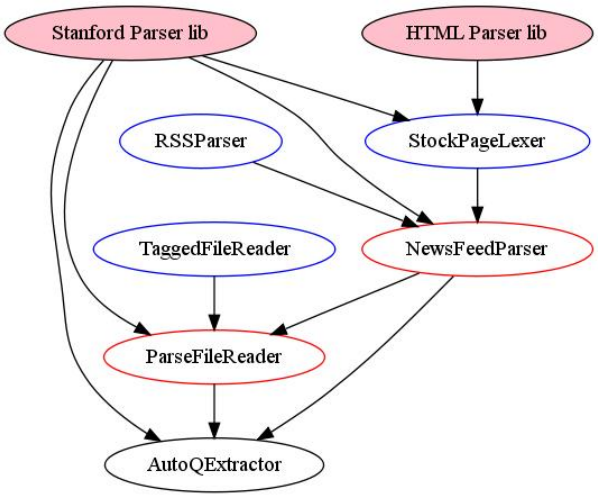

Figure 12: Dependency tree of the project

A brief description of the independent parts:

- **HTML Parser Lib, Stanford Parser lib** : Already available libraries.
- **StockPageLexer** : Extracts news from Yahoo (and MarketWatch) pages. Also, contains code for extracting tabular data.
- **RSSParser** : Fetches and parses Yahoo (and MarketWatch) files. Also, has scaffolding for a local RSS feed reader, which can provide updates locally instead of getting them off the Internet. Useful for testing purposes.
- **TaggedFileReader** : Parses the tagged files and provides a list of tags and the corresponding phrases in the news.
- **NewsFeedParser** : Connects the RSSParser and the StockPageLexer.
- **ParseFileReader** : Reads and annotates the sentences present.
- **AutoQExtractor** : Assimilates the results by running the parser over the manually tagged sentences and has code for running the analyser on new News articles.

The source code manager was bzr[11]. Also, most of the code is covered by the JUnit tests attached and is documented well in the *JavaDoc* format where ever necessary.

[11] http://www.bazaar.org

14 Conclusion and future scope of work

The system is solving successfully the primary problem that had been aimed at: namely, it is capable of extracting quantitative information in real time from natural language news, and, perhaps more importantly, is capable of filtering out news articles that have no quantitative content in them. Considering that the system had a very limited data set, which had been manually tagged, to rest its conclusions on, it provided us with a decent performance. However, there were a plethora of problems that were not meant to be solved here:

- **Named Entity/Time Recognition** : *For which a dictionary based method was employed.* We could, as a module, use an already existing NER for the purpose, for example. The Stanford NER itself. Besides, anaphora resolution is a current topic in research and work can be extended in that area too. Besides, the Stanford Parser itself makes available a Grammatical dependency graph for each sentence, which could resolve the subject for certain predicates, thereby giving more robustness.
- **Extraction of text from HTML pages** : *For which a rigid manual filter was employed.* Simple, as well as advanced generic methods of extracting text from HTML pages exist, for example the HTML density[12] method.
- **Knowledge base management** : For which one would have to keep a record of the news articles. There are academic as well as legal issues involved here. According the TOS of Yahoo! News, one is **not** to keep *databases* of information extracted from the content of their news in form of a database for *any* use. Nevertheless, keeping track of the information arriving from different sources would be a very interesting academic exercise. It can tell us about the reliability, spontaneity and accuracy of each source.
- **Finer classification** : Extracted news was at a very coarse level. However, the features that are used can be extended and trained for very precise form of classification of data. It is theoretically possible by concentrating on tense/syntactical function of the predicates (verbs) to extract sentences which contain only future estimates. The technique can very easily be extended to include the functioning of FASTUS[3], which specializes in filling templates for mergers and acquisitions by training on the verb-structure of the respective verbs.
- **Integrated trading** : If the quantitative information is reliable enough, then the system can be integrated with an Automated Trading system, as was our initial plan. This has an obvious speed advantage over a human interacting with the machine and then performing the trade. Though the system does effectively screen useless[12] pieces of news, and has been seen to yield profitable trades, an integrated system, though risky, would be better at harnessing the same.

One of the only problems faced while working on the project was the task of manually tagging the text. It was an arduous exercise, despite having the SentenceTagger GUI (figure 3) with us. In hindsight, the objectives changed drastically when the sparse nature of the data being collected was realised. Hence, before taking any step further in the project, volumes of more data needs to be tagged and processed.

References

[1] A. Soni, N. J. van Eck, , and U. Kaymak, "Prediction of stock price movements based on concept map information," *International Journal of Information Technology and Intelligent Computing*, 2006.

[2] S. L. Alonso, J. L. Bas, S. Bellido, J. Contreras, R. Benjamins, and J. M. Gomez, "Wp10 : Case study ebanking : Financial ontology," 2006.

[12] from a trader's point of view

[3] D. Israel, M. Kameyama, M. Stickel, and M. Tyson, "Fastus: A cascaded finite-state transducer for extracting information from natural-language text," in *IN ROCHE AND SCHABES (EDS.) FINITE STATE DEVICES FOR NATURAL LANGUAGE PROCESSING*. MIT Press, 1996, pp. 383–406.

[4] "Introduction to the conll-2005 shared task: Semantic role labeling."

[5] V. Punyakanok, D. Roth, and W. tau Yih, "The importance of syntactic parsing and inference in semantic role labeling," *Comput. Linguist.*, vol. 34, no. 2, pp. 257–287, 2008.

[6] D. McClosky, E. Charniak, and M. Johnson, "Reranking and self-training for parser adaptation," in *ACL-44: Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*. Morristown, NJ, USA: Association for Computational Linguistics, 2006, pp. 337–344.

[7] "Ptolemy ii." [Online]. Available: http://ptolemy.berkeley.edu/ptolemyII/

[8] "Stanford parser: A statistical parser." [Online]. Available: http://nlp.stanford.edu/software/lex-parser.shtml

[9] "Stanford named entity recornizer." [Online]. Available: http://nlp.stanford.edu/software/

[10] "Html parser library." [Online]. Available: http://htmlparser.sourceforge.net

[11] "Stanford manual annotation tool." [Online]. Available: http://nlp.stanford.edu/software/

[12] C. Kohlschütter and W. Nejdl, "A densitometric approach to web page segmentation," in *CIKM '08: Proceeding of the 17th ACM conference on Information and knowledge management*. New York, NY, USA: ACM, 2008, pp. 1173–1182.