Accurate Information Extraction for Quantitative Financial Events

Hassan H. Malik
hassan.malik
@thomsonreuters.com

Vikas S. Bhardwaj
vikas.bhardwaj
@thomsonreuters.com

Huascar Fiorletta
huascar.fiorletta
@thomsonreuters.com

Thomson Reuters
New York, NY, USA

ABSTRACT

In this paper, we present a novel financial event extraction system that achieves very high extraction quality by combining the outcome of statistical classifiers with a set of rules. Using expert-annotated press releases as training data, and novel feature generation schemes, our system learns multiple binary classifiers for each "slot" in a financial event. At runtime, common parsing and search indexing methods are used to normalize incoming press releases and to identify candidate event "slots". Rules are applied on candidates that satisfy a combination of classifiers, and the system confidence on extracted events is estimated using a unique confidence model learned from training data.

We present results of experiments performed on European corporate press releases for extracting dividend events, and show that our system achieves a precision of 96% and a recall of 79%.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*Retrieval models*; H.3.4 [**Information Storage and Retrieval**]: Systems and Software—*Question-answering (fact retrieval) systems*

General Terms

Algorithms, Experimentation, Performance

1. INTRODUCTION

Quantitative financial events such as corporate earnings announcements and revenue forecasts often serve as the primary driver for significant changes in asset prices in the financial markets worldwide. These events are typically delivered as unstructured text along with other information in corporate press releases over realtime news feeds. Structured delivery of important information pertaining to the events is preferred by professional investors and is also essential for algorithmic trading systems.

CIKM'11, October 24–28, 2011, Glasgow, Scotland, UK.

Considering that the precision of exisiting statistical systems is insufficient for practical use in the finance industry, financial information providers primarily rely on teams of professional analysts that constantly monitor news, and manually extract and publish events in a structured format.

Some highly customized, rule-based systems have also been developed to partially automate financial event extraction. However, these systems take a prolonged period of time to develop and require constant maintenance to accommodate for slight changes in the language of press releases.

In this paper, we present a novel event extraction system that achieves very high extraction quality by augmenting the outcome of multiple statistical classifiers with a set of rules. Since the bulk of the event extraction task is handled by the classifiers, and rules are merely used to handle exceptions, our approach significantly reduces the time needed to support new event types or to extend support of existing event types to new geographical markets. We also present a unique confidence estimation approach that classifies each extracted event into a set of predefined confidence categories.

2. RELATED WORK

Quantitative event extraction is an information extraction task that uses methods from information retrieval, machine learning, statistics and natural language processing along with other related research areas. It involves specialized pre-processing and normalization, entity recognition, relationship identification and confidence estimation. We refer the reader to [4] for a comprehensive survey.

Existing systems closest to our task are JASPER [1] and SCISOR [3]. Both of these systems are mainly rule-driven and use template-based pattern matching and other heuristic techniques. Our system uses a hybrid approach that combines statistical and rule-based methods and also assigns a confidence value to the extracted events. To the best of our knowledge, our system is the first such attempt in the domain of quantitative event extraction.

3. SYSTEM OVERVIEW

3.1 Terminology

We refer to the basic unit of information that is extracted from the unstructured press releases as a "fact". A fact, as we intend it, is a relevant financial event that our system is trained to detect and report. Common quantitative financial facts include corporate earnings, revenue, dividends, etc.

We may also think of fact extraction as a template filling

task. i.e., each fact may be considered as a template with a set of fields, some of which may be optional, and the task is to find suitable values for these fields from the unstructured content. We refer to each field in such template as a slot.

3.1.1 Learning Architecture

During the learning phase, a set of existing press releases are selected by expert analysts as training data for all target fact types. These press releases are first pre-processed to identify candidate facts and slots in each press release, and then presented to domain experts for annotation in a tool that was specifically developed for this purpose (Section 4.2). The annotated press releases are successively split in training and test sets used to construct the classification and confidence estimation models (Sections 4.3, 4.4 and 4.6). The resulting models and expert-written rules (Section 4.5) are then applied on the press releases in the test set in a way similar to the production phase. The results are subsequently evaluated by experts and additional iterations of this annotation/training cycle are made as needed.

3.1.2 Production Architecture

Our production system is organized as a multi-stage pipeline that receives XML requests containing the text of financial news enriched with some metadata. Each instance of the production system runs within a Jetty server, and scalability is achieved by concurrent stateless processing of multiple requests within each instance. Multiple production instances are used behind a hardware load balancer for fault-tolerance.

The first step in the production pipeline consists of preprocessing each press release with LUCENE and ANTLR for paragraph selection and tokenization respectively. The selected paragraphs are then analyzed in one pass over the data to identify candidate facts and slots, and features are extracted for each candidate to prepare instances for classification . The candidates are then classified using the statistical models obtained during the learning phase, and corresponding rules are applied on any positively identified candidates to handle known-errors, and to identify candidates that the classifiers are known to miss.

4. TRAINING THE SYSTEM

Given a set F of target fact types, a set T of training documents, and a set S of unique slot identifiers that contains all slots for facts in F, the training phase of our system begins with normalizing training documents.

4.1 Normalization and Tagging

Each document in T is normalized using an ANTLR-based tagging engine that uses a BNF grammar to identify synonyms such as "USD 0.05" and "5 US cents", and tokens that belong to the same category. For example Jan, January, and March all belong to the MONTH category. The normalized documents are then presented to analysts for annotation.

4.2 Annotation

The training data was annotated using a custom-developed tool that presents tagged documents to a human expert in a graphical user interface, and allows the user to identify and annotate the positive tokens for any slot in S from the set of candidates highlighted by the tagging engine (or to create tags that were missed by the tagging engine), while preserving the hierarchical nature of facts.

4.3 Feature Generation

The set of annotated facts is used to identify positive and negative examples for each slot where all slots of the same type but for a different kind of fact serve as negative examples.

For each example we then apply feature generation schemes on the marked up text that surrounds the target candidate in a window, the size of which is configurable. Our initial experiments have focused on extracting dividend and profit facts. For these fact types, we have experimented with many existing and novel feature generation schemes as shown below, and for each slot, selected a subset of these schemes based on our empirical evaluation (Section 6.1).

a) **Bag of Words (unigrams)**

b) **Delimiter-Present**: indicates delimiters occurring in the window

c) **Figure-Value threshold**: indicates if the numerical value of the slot is greater than pre-defined threshold

d) **Figure-Value-Log**: logarithm of the figure

e) **N-Grams**: bi/tri grams occurring in the window

f) **Distance-Farthest/Distance-Closest**: These schemes add a feature for each tag (word, phrase or normalized text) from a list of pre-defined tags for each slot type (selected based on domain knowledge) that occurs in the window. The feature value represents the spatial distance between the candidate slot and the matched tag. The Distance-Farthest scheme uses distance of the farthest instance of each matching tag as the feature value whereas the Distance-Closest scheme uses distance of the closest instance of each matching tag as the feature value.

g) **Before-Or-After**: This scheme adds a feature for each token/tag that occurs in a list of pre-defined tokens/tags. The feature is assigned a value of 1, 0 or -1 if the token/tag occurs after the candidate slot in the window, does not exist in the window, or exists before the candidate slot in the window, respectively.

h) **Period-in-Context**: This scheme applies to time-period-dependent fact types, and adds a feature with value = 1 if the time-period obtained from the document context (such as document title or metadata) matches the period specified in the window.

i) **Closest Single Matching Tag on Left / Right**: This scheme adds a feature indicating the single matching tag that occurs closest to the candidate slot, on its left or right, where the tag is taken from the same list as the Distance-Closest scheme.

After feature extraction we normalize values of all real valued textual features to the same scale by applying z-score standardization, and the resulting instances are then used to train the classification models for each slot.

4.4 Training and Combining Classifiers

We have used two different classification algorithms in our system, i.e., Linear SVMs and the Feature Weighting Classifier (FWC) [2]. Both of these algorithms are trained in linear-time and have been successfully applied to a variety of text classification problems.

Since the raw classification scores assigned to a particular sample does not accurately reflect the probability of the sample belonging to a particular class, we re-scale the

classifier scores using isotonic regression, which have been successfully used to obtain accurate class membership probability estimates for binary and multiclass problems [5]. We then combine the re-scaled scores of SVM and FWC using a weighted linear combination, where the weights were determined empirically, for each fact and slot-type.

4.5 Rules

In addition to using a combination of statistical classifiers, our system also incorporates a rule engine. Unlike existing rule-based systems, our system does not use rules as primary means for extracting the desired information from unstructured text, but instead uses rules to handle exceptions and to improve the overall system precision. In particular, the rules in our system aim to cover the following cases:

a) Handle rarely used verbiage or reporting standards, i.e., situations where high-precision classifiers could not be practically trained because of a lack of training examples.

b) Presence of outlier cases that are almost always incorrectly classified by the statistical classifiers.

c) Pruning certain types of samples from being classified. For example, our system uses a rule to exclude valid, but previously declared dividends (e.g., dividend for the same period last year) from being reported, to satisfy a business requirement.

4.6 Confidence Model

We also train a confidence model that is used to estimate and report the system confidence on each extracted fact. Our confidence estimation scheme focuses on measuring the textual similarity of an unseen press release against the training corpus. The confidence classes used in our system include HIGH, GOOD, MODERATE, and LOW. Confidence estimation allows users to act on the automatically extracted facts based on their tolerance to risks associated with acting on potentially incorrect information.

Our confidence model consists of a bi-gram corpus constructed from the annotated training set. All bi-grams that occur in windows surrounding each fact instance in the training set are added to this corpus, maintaining their frequency counts. This corpus is then used in the production phase to estimate the system confidence on extracted facts.

5. FACT FINDING

The real-time fact finding process consists of receiving a new document D as input, preprocessing the document and identifying candidates for each fact from a list of unique facts F and a unique slots S for each fact, classifying the candidates for each slot in S, applying relevant rules, assigning confidence and reporting the extracted fact. We now explain these steps in detail.

a) **Preprocessing** The incoming document is first indexed using Apache Lucene. Then for each fact in F, the document is queried with relevant keywords (identified by domain experts) to retrieve paragraphs that may potentially contain the fact.

b) **Candidate Selection and Feature Extraction** The selected paragraphs are normalized and tagged using the process explained in Section 4.1

c) **Classification and Classifier Combination** The candidate instances are then classified using the models trained in Section 4.4. Raw scores from SVM and FWC classifiers are normalized using isotonic regression. The normalized scores are then combined using the method explained in Section 4.4 and the resulting score is used to classify the candidate instance as positive or negative.

d) **Applying Rules** Depending on the fact and slot type of a positively classified instance, the rule engine is optionally invoked in order to prune common errors, and to handle the other situations explained in Section 4.5.

e) **Computing Confidence Scores**

We finally estimate confidence on each extracted fact. We use the normalized window text to create a corpus of bigrams B. The confidence score is then calculated as follows:

$$ConfidenceScore = \frac{\sum_{b \in B} counts(b)}{|B|}$$

where $counts(b)$ indicates the number of times the bigram b appears in the training corpus (section 4.6). Various thresholds that use mean and standard deviation of window scores in the training corpus are then applied to map the confidence score to a confidence class (Table 1).

Condition	Confidence			
	HIGH	GOOD	MODERATE	LOW
If score >	$\mu + 2\sigma$	$\mu + \sigma$	μ - σ	otherwise

Table 1: Thresholds for confidence assessment, μ is mean and σ is standard deviation of the training corpus scores

6. EMPIRICAL EVALUATION

Our dataset consisted of English financial press releases for European companies from January 2006 to May 2010. Our experiments focused on Dividend and Profit facts; selected based on business priorities in our organization.

We first compare various feature generation schemes and classifier combination methods on the main slots for each fact type, i.e., the dividend and profit figure slots, using a 10-fold cross-validation on the annotated data. Note that these experiments are limited to the main slots, and no rules are applied at this stage. Additional slots must be obtained before the fact is considered to be complete and publishable. Section 6.3 evaluates our system on complete facts.

6.1 Comparing Feature Generation Schemes

In this section we compare the feature generation schemes discussed in Section 4.3. We used Bag-of-words as our baseline scheme and measured the incremental improvement achieved by each feature generation scheme, when combined with bag of words. Tables 2 and 3 present the results of this experiment for the dividend and profit figures, respectively. We observe that not all schemes are effective for all slots. Therefore, the final classifiers for each slot were constructed using a subset of feature generation schemes that yielded at-least some improvement over the bag-of-words baseline. For example, the precision and recall results in the last row of Table 3 were obtained by using all schemes to generate features, except "Period In Context".

6.2 Classifiers and Classifier Combination

In this section we compare the performance of SVM against that of FWC. We performed a 5-fold cross validation on the

Features	Precision	Recall
BOW	0.94	0.95
BOW+nGrams	0.96	0.97
BOW+Before-Or-After	0.97	0.96
BOW+Period-in-Context	0.94	0.95
BOW+Delimiters	0.95	0.95
BOW+Distance	0.96	0.94
BOW+ClosestWordLeft	0.97	0.96
BOW+Figure-Threshold	0.95	0.93
BOW+Figure-Value-Log	0.94	0.95
ALL	0.98	0.97

Table 2: The performance of feature generation schemes on the Dividend-Figure

Features	Precision	Recall
BOW	0.86	0.87
BOW+nGrams	0.87	0.88
BOW+Before-or-After	0.88	0.87
BOW+Period-In-Context	0.85	0.87
BOW+Delimiters	0.86	0.88
BOW+Distance	0.91	0.92
BOW+Figure-Threshold	0.90	0.88
BOW+Figure-Value-Log	0.92	0.91
BOW+ClosestWordLeft	0.91	0.90
ALL	0.97	0.96

Table 3: The performance of feature generation schemes on the Profit-Figure

dividend-figure slot. From Table 4, we observe that SVM outperforms FWC in terms of precision, whereas FWC outperforms SVM in terms of recall, thus motivating us to combine the outcome of these methods.

Set	SVM		FWC	
	Prec	Recall	Prec	Recall
1	0.98	0.94	0.83	0.97
2	0.98	0.94	0.82	0.97
3	0.99	0.92	0.80	0.98
4	0.98	0.95	0.79	0.99
5	0.99	0.95	0.84	0.99

Table 4: Comparing SVM and FWC classifiers (for the Dividend-Figure)

As we have explained in Section 4.4, our system computes the final classification score as a weighted linear combination of normalized individual classifier scores. To determine weights for the classification methods, we applied 10-fold cross validation on training data for dividend and profit figures, and evaluated three different weight combinations. Table 5 presents the results of this experiment. We observe that the best performance is achieved when SVM and FWC are assigned 70% and 30% weight, respectively. Therefore, we used these weights in the rest of our experiments.

6.3 Overall System Performance

All the results presented so far cover a single slot. It is therefore important to evaluate the overall system performance when a fact is obtained by combining several slots, each with its own classifier and rules.

For this purpose, we applied our system on a new set of 604 press releases from a period of 16 months, not all of which contained facts. Our team of annotators manually verified the system output and inspected the press releases for additional unreported facts. Table 6 presents the results of this experiment. Our system achieved a precision of 96% and a recall of 79%, when the facts were classified as high or good confidence. It is important to note that the system identified many additional facts in lower confidence categories but these facts are not included in the system output because of high-precision requirements in our domain.

Combination Method	Dividend		Profit	
	Prec	Recall	Prec	Recall
Linear SVM_70 FWC_30	0.98	0.97	0.97	0.96
Linear SVM_50 FWC_50	0.97	0.95	0.98	0.97
Linear SVM_30 FWC_70	0.94	0.95	0.91	0.90

Table 5: Different classifier combination methods for dividend and profit

Facts				
Extracted	Good/High Conf.	Correct	Actual	Missed
454	414	398	503	82

Precision	Recall
0.96	0.79

Table 6: Overall system performance for Dividend Facts

7. CONCLUSIONS AND FUTURE WORK

We have presented a novel event extraction system that uses a combination of multiple binary classifiers and manually written rules, where the bulk of the extraction task is handled by the statistical classifiers and rules are used to handle exceptions. We also present a unique approach to estimate system confidence on each extracted fact.

In the future, we plan to extend our system to support additional quantitative fact-types such as revenue and earnings per share, and non-quantitative events such as mergers and acquisitions.

8. REFERENCES

[1] P. M. Andersen, P. J. Hayes, A. K. Huettner, L. M. Schmandt, I. B. Nirenburg, and S. P. Weinstein. Automatic extraction of facts from press releases to generate news stories. In *Proceedings of the third conference on Applied natural language processing*, ANLC '92, pages 170–177, 1992.

[2] H. Malik, D. Fradkin, and F. Moerchen. Single pass text classification by direct feature weighting. *Knowledge and Information Systems*, pages 1–20, 2010. 10.1007/s10115-010-0317-9.

[3] L. F. Rau and P. S. Jacobs. Integrating top-down and bottom-up strategies in a text processing system. In *Proceedings of the second conference on Applied natural language processing*, ANLC '88, pages 129–135, 1988.

[4] S. Sarawagi. Information extraction. *Found. Trends databases*, 1:261–377, March 2008.

[5] B. Zadrozny and C. Elkan. Transforming classifier scores into accurate multiclass probability estimates. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '02, pages 694–699, New York, NY, USA, 2002. ACM.