BUILDING SEMANTIC-RICH PATTERNS FOR
EXTRACTING FEATURES FROM EVENTS OF
AN ON-LINE NEWSPAPER

Ray R. Hashemi, Charles Ford, Jr., Ashish Bansal, Stephen D. Sieloff, and John R. Talburt

1Department of Computer science  2Department of Computer science  3Acxiom Corporation
Armstrong Atlantic State University  University of Arkansas at Little Rock  11 Technology drive
11934 Abercorn Street  2801 South University Avenue  Little Rock, Arkansas, 72211
Savannah, GA 31419, USA  2 Little Rock, AR 71204, USA  USA

ABSTRACT
To extract features from an event in an on-line newspaper, two major steps are involved: Extracting the free formatted events' announcements from the web sites and (b) extracting the features from the announcements. We have completed the first step and have reported our findings in [4]. The exploration of the second step is the concern of this research effort. This exploration leads to the introduction and development of a pattern-based technology. Which differs from Frame Technology and XML technology. The major component of this technology is a body of semantic-rich patterns obtained from subsets of announcements by dissecting and extracting features of interest. A methodology for systematically building such patterns with minimum effort is introduced in this paper. Also, preliminary results supports the fact that the pattern-based technology is a powerful vehicle.

KEYWORDS
Web Mining, Event Analysis, Ontology, Pattern Generation, and Clue Finding.

1. INTRODUCTION

An on-line newspaper reports many events. An event is a short and data-rich document such as announcements for "wedding", "birth", "graduation", "cars", "auction", "obituary", "divorce", etc. Extracting data from an event is highly desirable for the purpose of collecting, selling, comparing, and summarizing data, as well as decision making. For example, extracting the following features from an "auction" announcement has great value to an individual with interest in antiques: Location, day, and time of the auction; type and age of auctioned materials; name of the auctioneer, etc. The extracted features from an Obituary announcement may be used for fraud prevention, taxation, bankruptcy, inheritance, etc.

In extracting features from an event of an on-line newspaper, two major steps are involved: Extracting a set of event announcements from a web site and (b) extracting the features from the announcements. We have completed the first step and have reported our findings in [4]. The exploration of the second step is the focus of this research effort.

An event announcement may follow either a fixed or a free format. Each fixed formatted announcement follows a pre-set template whereas those with free format do not follow any format at all. Extracting features from a fixed formatted announcement is a trivial process so we will not pursue it further. However, extracting features from a free formatted announcement is a challenge and hence will be explored in this paper.

The number and types of features that a free formatted announcement carry are usually unknown. They may vary from one announcement to the next within the same event and for the same on-line newspaper. As a result, we need an architecture that not only extracts features from an announcement, but also extracts the relationships among the features.
One such architecture is the Frame Technology [1, 2, 3]. The major shortcoming of this methodology is lack of flexibility. That is, the number of frames and slots within each frame must be well defined in advance. This \textit{a priori} requirement makes this architecture less suitable to support our goal.

A new and emerging technology that removes the shortcomings of the Frame technology is the set of technologies collectively known as eXtensible Markup Language (XML). This technology includes XML, Xpath, Xlink, XSL, XSLT, Xquery, and Xpointer [5, 6, 7, 8]. XML describes a document structure and meaning by allowing the author to design and develop his/her own set of tags to capture a feature and the semantic of the feature. The main disadvantage of XML is that one can take advantage of XML benefits only after the announcement is tagged. In other words, the process of tagging must be repeated for every announcement. Unfortunately, previously tagged announcements do not expedite this process.

As an alternative, we introduce a \textit{pattern-based} architecture. In this architecture, we employ semantic-rich patterns for extracting features and preserving their relationships. A semantic-rich pattern is a pattern that supports the semantics both at the feature level and the announcement level. These Patterns are built out of the sentences of the announcements. A pattern generated from one announcement may be applied to the next. For analysis of any new announcements, only a subset of patterns that correspond to the contents of the announcement may be used to dissect and extract features and their relationships from the announcement. The flexibility of the architecture lies in the fact that for any free formatted announcement, there is always a set of patterns that can explore the announcement. In contrast to XML technology, patterns learned from one announcement may be used to extract features from another announcement that have not been encountered before.

The major component of the pattern-base technology is a body of semantic-rich patterns. A methodology for building such patterns systematically with minimum effort is the documented in this paper. In the future, XML may be used for tagging of announcements from the generated patterns.

To show the effectiveness of the methodology for building semantic-rich patterns, we concentrate on only one type of the event, namely, Obituary. The methodology is introduced in section 2 and the preliminary findings are the subject of section 3. The conclusion and future research are covered in section 4.

2. METHODOLOGY

In this section we first describe all the possible features of interest and their relationships within the text of an Obituary announcement using ontology and followed by a description of how to use the ontology to generate patterns.

2.1 Features of interest

Features of interest within an Obituary announcement may be presented by the ontology of Figure 1. In this paper, we refer to every symbol (rectangle or oval shape) in Figure 1 as a “node”. For the sake of space, time and simplicity, we focus only on a sub-section of this ontology shown in Figure 2.

In Figure 2, we have also added a parenthesized name to each node. (For simplicity, Figure 1 does not show the parenthesized names.) Close inspection of nodes in Figure 2 reveals that for some nodes the parenthesized names are specified using upper case letters while others are in lower case. Nodes whose parenthesized names are in upper case represent features of interest. It should be clear then that the nodes “Residence”, “Previous”, and “Current” are not features of interest. One may ask how one can make a distinction between a previous city of residence and a current city of residence, for example. This semantic has been captured in the parenthesized name for “City” by using prefixes of “P” and “C” as part of the parenthesized name for the City. Parenthesized names, also called “structural tokens”, are fully described in the next sub-section.
2.2 Pattern builder (PB)

The pattern builder acts on two major ingredients, namely: features of interest and their relationships. Each generated pattern corresponds to a sentence in a given announcement and accentuates those components of the sentence that serve as features of interest.

A pattern acts as a template for future use. For this very reason, the pattern must have the four properties of being:

- self-explanatory,
- able to preserve the semantics at the feature level,

Figure 1. Features of interest for an obituary announcement using ontology.
able to preserve the semantic at the announcement level, and
represented in a compact form.

To provide for these properties, we introduce (a) a set of tokens that can capture the semantics of the sentence component for which they stand and (b) a mechanism for the integration of the tokens with the ontology to capture a higher level of semantics defining the relationships among the tokens.

2.2.1 Tokens

The pattern generated by PB is made of a set of tokens. These tokens are divided into four groups, namely, Primitive, Structural, Object, and Supper. A Primitive token is a sequence of characters separated from the previous and next token by a space or a punctuation mark. The number of characters in the primitive token is referred to as the length of the token. The primitive tokens are:

- **C** is composed of alphabets, starts with a capital letter and its length is greater than one.
- **L** is a C token for which the length is one.
- **N** is a token consisting of all digits.
- **B** is a token composed of a sequence of alphabets, not starting with a capital letter, where its length is greater than or equal to one.
- Any punctuation mark is a token and represents itself as a primitive token (e.g. C.C represents three tokens of “C”, “.”, and “C”).

A Structural token is a two-character code that represents a sequence of primitive tokens at a higher level of abstraction. They are:

- **AG** is an N token (an Age)
- **NM** is a mixture of C, L, and possibly punctuation marks (a Name).
- **BD** is a mixture of C, N, and punctuation marks (a Birth Date).
- **BP** is a mixture of C and possibly punctuation marks (a Birth Place).
- **DD** is a mixture of C and possibly punctuation marks (a Death Date).
- **DP** is a mixture of C and possibly punctuation marks (a Death Place).
- **CC** is a mixture of N and possibly punctuation marks (a Current City of residence).
- **CS** is a mixture of C, L, and possibly punctuation marks (a Current State of residence).
- **CZ** is a mixture of N and possibly punctuation marks (a Current address’s Zip-code)
PC is a mixture of C, L, and possibly punctuation marks (a Previous City of residence)
PS is a mixture of C and possibly punctuation marks (a Previous State of residence)
PZ is a mixture of N and possibly punctuation marks (a Previous address’s Zip-code)

An Object token is a code made of two capital letters to represent objects (i.e. the people who appear in an announcement) and their relationships. Figure 1 shows that the number of needed object tokens is 198. The possible combinations of two letter codes are large enough to represent all the people that may appear in an obituary announcement. Some of the objects and their tokens are:

DE (The Deceased), SP (Spouse of the deceased),
SO (Son of the deceased), OW (wife of SO), OS (son of SO), OD (daughter of SO),
DA (Daughter of the deceased), AH (Husband of DA), AS (Son of DA), AD (Daughter DA),
SN (Step Son of the deceased), NW (wife of SN), NO (son of SN), ND (daughter of SN),
SE (Step daughter of the deceased), EH (Husband of SE), ES (Son of SE), ED (Daughter SE),
BR (Brother of the deceased), BW (Wife of BR), BS (Son of BR), BD (Daughter BR),
SI (Sister of the deceased), SH (Husband of SI), SS (Son of SI), SD (Daughter of SI),
PB (Step Brother of the deceased), PW (Wife of PB), PS (Son of PB), PD (Daughter PB),
PT (Step Sister of the deceased), TH (Husband of PT), TS (Son of PT), TD (Daughter of PT),

A Supper token is a four letter code composed of an object token followed by a structural token. For example,

DENM (The name of the deceased), FFNM (The name of the deceased's father), SONM (The name of the deceased's son), SPNM (The name of the deceased's spouse), BRNM (The name of the deceased's Brother), BWNM (The Name of the Brother's wife of the deceased), BSNM (The Name of the Brother's son of the deceased), DANM (The deceased's daughter Name), SONM (The name of the Son of the deceased), OSNM (The name of the deceased son of the son), GSNM (The name of the deceased son of the son),

2.2.2 Integration of the tokens and the ontology

Integration of the tokens together with the ontology enables us to capture a higher level of semantics among the tokens. To smooth such integration we built a GUI to provide a systematic approach for generating the patterns. The GUI is composed of three sections to describe “event’s ontology”, display “event’s announcement”, and display “intermediate results”. Any part of the announcement may be highlighted, dragged, and dropped into a node within the ontology. The Pattern Builder, PB, dissects those nodes of the ontology that are updated by a drag-and-drop mechanism and uses the following algorithm to automatically and systematically build a pattern that satisfies the four desired properties for a pattern that was mentioned previously.

Algorithm Pattern-Generator

Given: An announcement made of a set of sentences \(\{s_1, \ldots, s_n\}\), an ontology, and a GUI that presents both the announcement and the ontology along with a display of intermediate results.

Objective: Build a pattern from any sentence, \(s_i\), within the announcement.

Step 1. //convert \(s_i\) to its primitive tokens.

- Generate an image of the sentence \(s_i\). (Image is the representation of \(s_i\) in its primitive tokens.)

Step 2. //Impregnate the ontology nodes.

- Repeat step 2 while there is a feature of interest in \(s_i\) that has not been identified.

  - Highlight, drag-and-drop a feature of interest into the ontology. (A feature may be made of one or a set of sequential words.)
  - The corresponding tokens in the image of the sentence are replaced by the super token that represents the feature.
  - Add the corresponding primitive tokens of the highlighted feature to Primitives Table.

Step 3. //Impregnate the “clue” button within the GUI by clues within \(s_i\) that has not been identified.

- Repeat step 3 while there is a clue of interest in \(s_i\).

  - Highlight, drag-and-drop the clues of interest into the “clue” button presented within the GUI interface. (A clue may be made of one or a set of sequential/non-sequential words. GUI has a provision for identifying the clues made of non-sequential words.)
Add the highlighted clue along with an assigned clue identification number (Clue-Id) to the Clues Table.

The primitive tokens in the image of the sentence that are corresponding to the identified clue are replaced by Clue-Id (the special token of four digits.)

Add the first word of the clue, Clue-Id, and the number of non-contiguous segments of the clue (# of chunks) in the Key-Words Table.

Step 4. //Cleaning the image of sentence

Replace the remaining primitive tokens of the image that appear between the first non-primitive token and the last non-primitive token in the image with symbols [%%].

Replace the remaining primitive tokens of the image with symbols [!!].

Step 5. Add the outcome of Step 4 as a pattern to the Pattern Table.

Step 6. End.

The following example is used to demonstrate how the algorithm works.

**Example:**

Given the sentence, $s_i$:

$$s_i: \text{Dr. John M. Doe of St. Louis who was the Director of Mercy Hospital died on Tuesday, July 4, 02 in a car crash near Boston.}$$

The image of the sentence is created in Step 1:

Image:  C.C L.CBC.CBBBCBCCBBC,CN, NBBBBBC

As a part of Step 2, the feature “John M. Doe” is highlighted, dragged and dropped into the Name node of the Deceased, and the image of $s_i$ changes into:

Image:  C.[DENM]BC.CBBBCBCCBBC,CN, NBBBBBC

The primitive tokens of CL.C are added to the Primitives Table. We continue with highlighting and dragging and dropping the features of “St. Louis”, “Tuesday, July 4, 02” into their proper nodes of the ontology. The image of sentence changes into:

Image:  C.[DENM]BC.CBBBCBCCBBC,CN, NBBBBBC

Applying Step 3 causes that the clue of “of . . .died on” to be highlighted, dragged, and dropped into the “clue” node of the GUI. The corresponding primitive tokens in the image of the sentence are replaced by the Clue-ID = 0001. In addition the clue along with Clue-Id are added to the Clues Table. The first word in the clue, “of” is added to the Key-Words Table. The image of the sentence after applying step 3 is now:

Image:  [!!][DENA][0001][DECC][BCCB][0001][DEDD][BBBBBC]

The result of applying Step 4 is:

Image:  [!!][DENA][0001][DECC][%][0001][DEDD][!!]

Step 5 expresses that the above image is the pattern generated for statement $s_i$. This pattern is added to the Patterns Table along with an assigned Pattern-Id. The pattern-Id for the pattern is also propagated to the clues in the Clues Table while key words are used to update the Key-Words Table for those clues and key words that are extracted in the process of generating the pattern. The reader should be reminded that the Patterns Table contains no duplicate patterns. A sample of each of the four tables is shown in Figure 3.

3. RESULTS

We collected 866 Obituary announcements from 68 on-line newspapers. All announcements were free formatted. We generated 50 patterns out of the 173 randomly selected announcements (20% of the total number of announcements) as a training set. The remaining 693 announcements served as a test set. Out of the 693 announcements the system was unable to extract all the features from 16 of them (i.e. 677 announcements are totally analyzed by the system.) Also, among the totally analyzed announcements, the extracted features for 17 of them are incorrect.

To evaluate the performance of the system, we introduce the two ratios of Completeness and Accuracy which are defined using Formulas 1 and 2, respectively:
Completeness = \( \frac{\text{Card}(G_o)}{\text{Card}(G_t)} \) \hspace{1cm} (1)
Accuracy = \( \frac{\text{Card}(G_o - G_i)}{\text{Card}(G_o)} \) \hspace{1cm} (2)

where,
\( G_o \) is a subset of announcements in the test set that can be totally analyzed by the set of generated patterns (i.e. all the existing features in every announcement are extracted),
\( G_t \) is the test set, and
\( G_i \) is a subset of announcements in \( G_o \) for which at least one extracted feature is incorrect.

The completeness and accuracy ratios for our test set were 97.6\% and 97.4\%, respectively.

4. CONCLUSIONS AND FUTURE RESEARCH

The accuracy and completeness ratios reveal that the pattern-based methodology for extracting features of interests and their relationships from an Obituary announcement is promising. The methodology is flexible enough to support the extraction of features from one free formatted announcement to the next and powerful enough to capture internal and external semantics of the features.

One may ask why the patterns generated by the above algorithm are semantic-rich. To answer the question, let us examine the following pattern:

\[[004][DENA][0004][DECC][%][002][DEDD][!!]\]

The super token DECC, for example, determines that the corresponding text to DECC, in the announcement is a “name of city”, at the feature level and the text is also the “current city in which the deceased lived”, at the announcement level. The feature level and the announcement level semantics are hidden only in the four-letter token of DECC. That is, the pattern is also compact and self-explanatory. It is clear that the patterns generated by our methodology have all four proposed properties.

As future research, two areas of interest are under consideration: (a) Testing the methodology for a high volume of announcements and (b) Creating usable records out of the extracted features by finding and augmenting data that possibly exist in other sources available to the authors.

![Figure 3. A sample of tables created by the Patter-Generator Algorithm, (a) Super tokens Table, (b) Clues Table, (d) Patterns Table, and (c) Key-Words Table.](image)
REFERENCES

8. www.w3.org (The Worldwide Web Consortium Extensible Markup Language homepage.)