Applying Similarity Measures for Management of Textual Templates

Cécile Boisson
Department of Computer and Information Science
Linköpings universitet, S-581 83 Linköping, Sweden
e-mail: cecbo@ida.liu.se

Abstract

We present an approach to template generation, where each template represents a set of "similar" text patterns. We describe the characteristics of the templates and a general task of template management, where syntactical similarity measures are used to efficiently generate and generalize templates. We also present an algorithm and a prototype implementation in a journal editing application. Test results show the importance of combining language dependent and independent syntactical measures.

1 Introduction

There are many applications where a piece of text or an utterance triggers an operation. This is the case in domains such as information extraction, spoken dialogues, and intelligent publishing:

- In information extraction systems which contribute to the task of Message Understanding, pieces of text trigger the instantiation of "templates" which describe events or facts relevant to a given task. See for example (Cowie and Lehnert, 1996), (Lehnert et al., 1993), (Gaizauskas et al., 1995) and (Hobbs et al., 1996).

- In spoken dialogue applications such as (Strömbäck and Jönsson, 1998) or (Qvarfordt and Jönsson, 1998), user utterances trigger the generation of an answer.

- In professional electronic publishing applications, editorial instructions are established to define a uniform editing style. Some of them can be based on the content of the text to be published. In the case of an electronic journal which publishes scientific articles, such an instruction specifies that any reference to an article, e.g. "`in the IJCAI paper of Harris`", has to be transformed into a link to the actual article.

Many of these tasks can be repetitive and thus do the same kind of operations again and again:

- An IE system can be applied on new corpora.

- A spoken dialogue system answers each new utterance.

- The editorial instructions of a journal are applied for the editing of each issue.

These applications share three characteristics. First, certain patterns (i.e. pieces of text or utterances) trigger operations. Second, some of the patterns trigger the same operation. Finally, an automation of the identification of the patterns and of the execution of the operations would save time and resources. Our work aims at providing this automation.

In the remainder of this paper all our examples are taken from a real application in intelligent publishing for an electronic journal.

The automation of the identification of patterns implies the learning of rules of the form (pattern, operation) such as:

rule 1: ("`in the article`",
`create-link-to-article()`)

rule 2: ("`my article`",
`create-a-link-to-article()`)

rule 3: ("`in the IJCAI paper of Harris`",
`create-a-link-to-article()`)

rule 4: ("`in this mail`",
`create-a-link-to-mail()`)

Note that subsets of the set of rules trigger the same operation. Representing all the patterns that trigger the same operation in a few readable expressions, would ease the presentability of the set of rules. For example, "`(in) [the|my] article`" can identify the patterns of rule 1 and rule 2 that trigger the same operation `create-a-link-to-article()`. Moreover, replacing information in the patterns with more general information provides a larger coverage of the set of patterns that trigger the same operation. A method which extends the coverage can provide a more effective automation in terms of recall. For example: "`(in) <determiner> article`" can identify "`the article`" and also "`this article`" but not "`in this mail`". However, to keep an acceptable level of precision, such a method should also provide means to minimize the risk of irrelevance.

We introduce the term *template* for a structure that can be used to identify a set of patterns that trigger the same operation. Templates management is an important requirement for the automation of the identification of patterns. The management consists of creating and keeping up to date a set of patterns that is (1) general enough to identify as many patterns as possible, (2) specific enough to identify the correct patterns only, and (3) simple enough to ease the readability of the set of rules. We measure these three characteristics in terms of recall, precision and number of templates[1].

In this paper we apply syntactical similarity measures to generalize similar templates into a unique template.

There are several possible approaches to measure syntactic similarity. We look at two categories of measures: *grammar based* and *word based*. In the first category, a grammar specifies rules to replace or delete pieces of a template. Two templates that are rewritten into the same "form" are similar. Word based methods such as the *edit distance*, are widely used in information retrieval. The edit distance between two strings is the minimum number of [token] insertions, deletions and replacements needed to make them equal (Baeza-Yates and Ribeiro-Neto, 1999). We assume that a token is a word, a concept or a punctuation mark. Two strings are similar if the edit distance is lower than a given threshold. Writing a grammar implies that some kind of syntactic rules exist in the language used for expressing patterns, while giving a threshold for the edit distance is independent of the language. We compare, in terms of number of templates, precision and recall (Baeza-Yates and Ribeiro-Neto, 1999), the set of templates that are generated with each category of similarity measures. We propose to exploit the best of two worlds by combining the two measures.

In section 2 we describe the task scenario from which most of the examples of this paper are inspired, and that we use as a test platform for our testing. In section 3 we present the general template management task and introduce the various similarities that are the basis of our algorithm. In section 4 we give our algorithm to generalize syntactically similar templates. In section 5 we test our algorithm and analyze the results. In section 6 we relate our approach to other work. The paper is concluded in section 7.

[1]Note that the choice of minimizing the number of templates (and thus the number of rules per operation) should be balanced by the level of readability of the new templates.

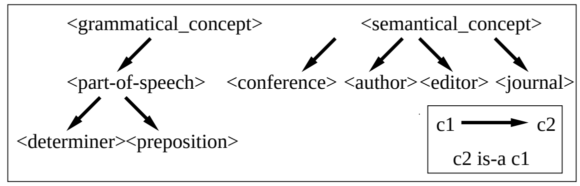

Figure 1: Ontology for ETAI

2 Test Platform

We identified the need for support for pattern identification in the task of the editor of ETAI (ETAI, 1997), an electronic journal for artificial intelligence. While our method is adaptable to other domains, it is on this task that we have performed the evaluation of our methods of template management. Most of the examples which illustrate our explanations are also examples from this task. The policy of the journal includes the public and collaborative reviewing of the articles submitted for publication. The task of the editor is to compile the email messages written by researchers to review a given article, into a web page. Each message is processed in its order of arrival. The messages can contain references to articles. One of the operations is to transform the pieces of text that refer to articles into links to the actual articles. Thus in this task, the patterns to recognize are the references to articles, and the texts to inspect are bodies of messages. Examples of patterns are "`in this paper`" and "`in the IJCAI paper of Harris`". The ontology for this task is described in figure 1. Each word of the patterns is associated to a `<part-of-speech>`. Some words, alone or in sequence, are also associated to a `<semantical-concept>`.

3 The Task of Template Management

We want to define a management of the set of templates which eases the presentability of the set of rules and provides a large but precise identification of the patterns in the texts processed.

Figure 2 defines the requirements of our management task for a set of templates in terms of input and output. The idea is to take a set of

Input:
- T: a set of templates.
- C: a set of concepts.
- m: a measure method.
- g: a generalization method.

Output: T' a set of templates such that:
1. T' covers at least all the patterns covered by T, i.e. $\bigcup_{t_j \in T} P_{t_j} \subseteq \bigcup_{t'_k \in T'} P_{t'_k}$.
2. For each template in T' there is a similar template in T with respect to the measure m, i.e.
   $\forall t'_a \in T', \exists t_b \in T :$ `areSimilar`(t'_a, t_b, m).
3. T contains at least as many templates as T', i.e. $|T| \geq |T'|$.

Figure 2: Management Task for a Set of Templates

templates as input, process it and provide a new set that can identify at least as many patterns and which is smaller or equal in size.

The task is based on the measurement of the similarities between the templates of the set. A template is composed of four features described in figure 3. We measure the similarity of two templates according to the similarity between their features. We identify four categories of similarities, one for each feature. They are described in figure 4.

The following illustrates the task of the management of a set of templates for an application of electronic editing. In this example, we adopt the notation T_{o_i} for the set of templates that trigger the operation o_i.

The input of the task is composed of:

- C: a set C of concepts.
  C = {`<author>`, `<determiner>`}

- m: the measure method. It computes the syntactical similarity with a grammar on the matchers of the templates that trigger the same operation. Thus the syntactic similarity is computed only between templates belonging to the same set T_{o_i}. In this example, the grammar rewrites the tokens with their *part-of-speech* tag. Thus matchers with the same *part-of-speech* structure are similar. Moreover the grammar rewrites the *part-of-speech* tags <determiner> and <preposition> with the empty token. Thus matchers can be similar even if they contain a different number of determiners and prepositions.

- g: the generalization method which replaces pieces of the matcher with concepts from C.

A template t_i is composed of four features:
1. s_{t_i} the set of objects from which t_i is generated.
   We call these objects *generators for* t_i.
   s_{t_i} can contain patterns or templates.
2. o_{t_i} the operation that is triggered by p_{t_i}.
3. m_{t_i} the matcher: a sequence of tokens that is used to identify patterns, where a token is either a word, a punctuation mark or a concept.
4. P_{t_i} the set of patterns that t_i can identify.

Example: given the pattern p such that
p = "`the results by Hill and Smith`",
$t_1 = (s_{t_1}, o_{t_1}, m_{t_1}, P_{t_1})$ where :
s_{t_1} = {p},
o_{t_1} = "`create-a-link-to-article()`",
m_{t_1} = "`the results by <author>`
`and <author>`",
P_{t_1} = {p: p = "`the results by A and A`",
where:
A = `anyValueFromConcept(<author>)`}

Figure 3: Structure of a Template: Four Features.

1. $s_{t_i} \cap s_{t_j} \neq \emptyset$.
   t_i and t_j are similar because they share some generators.
2. $o_{t_i} = o_{t_j}$.
   t_i and t_j are similar because the patterns identified by t_i and t_j trigger the same operation.
   We say that t_i and t_j are operation similar.
3. `areSyntacticallySimilar`$(m_{t_i},\ m_{t_j})$.
   t_i and t_j are similar because their matchers are syntactically similar.
   We say that t_i and t_j are syntactically similar.
4. $P_{t_i} \cap P_{t_j} \neq \emptyset$.
   t_i and t_j are similar because some of the patterns identified by t_i are also identified by t_j.

Figure 4: Templates Similarities.

For example, if `Harris` is an `<author>` then "`the paper from Harris`" will be generalized to "`the paper from <author>`"

- T: a set of six templates t_1, t_2, t_3, t_4, t_5 and t_6 such that t_1, t_2, t_3 and t_4 trigger the same operations o_1, t_5 triggers another operation o_2 and t_6 triggers a third operation o_3.
  We note: $\mathrm{T} = \{t_1, t_2, t_3, t_4, t_5, t_6\} = T_{o_1} \cup T_{o_2} \cup T_{o_3}$ where $T_{o_1} = \{t_1, t_2, t_3, t_4\}$, $T_{o_2} = \{t_5\}$, $T_{o_3} = \{t_6\}$.
  Each template t_i has a matcher m_i as follows:
  m_{t_1} = "`in the article`"
  m_{t_2} = "`my article`"
  m_{t_3} = "`this paper`"
  m_{t_4} = "`the paper by Harris and Smith`"
  m_{t_5} = "`send me a copy`"
  m_{t_6} = "`a = b+c`"

The computation of the similarity discovers that: t_1, t_2 and t_3 are similar and that we can rewrite them in a unique template t_7 whose matcher is "`(in) [the|my|this] [article|paper]`".

Moreover, the generalization g produces the following rewriting:

- g(m_{t_4}, C) = $m_{t'_4}$ = "`the paper by <author> and <author>`"
- g(m_{t_5}, C) = $m_{t'_5}$ = "`send me a copy`"
- g(m_{t_6}, C) = $m_{t'_6}$ = "`a = b+c`"
- g(m_{t_7}, C) = $m_{t'_7}$ = "`(in) <determiner> [article|paper]`"

Thus, T' is a new set of templates such that T' = $\{t'_4, t_5, t_6, t_7\}$ and T' = $T'_{o_1} \cup T'_{o_2} \cup T'_{o_3}$ where $T'_{o_1} = \{t'_4, t_7\}$, $T'_{o_2} = T_{t_5}$ and $T'_{o_3} = T_{t_6}$.

In the next section, we give the algorithm that performs this combination of grouping the similar templates and generalizing by replacing specific information with concepts.

4 An Algorithm for Template Management

Figure 5 presents our algorithm for template management that generates a template from a given pattern and integrates the new template in the set of templates. This algorithm is executed for each new pattern p to learn.

First the pattern is generalized into a template t. The instruction, "$\texttt{patternGeneralization}(p, C)$", generates the matcher of a new template by replacing specific information in the pattern p with more general information (i.e. concepts). For example, given the semantical concept "`<author>`" and a set of *part-of-speech* tags as grammatical concepts, the pattern "`the results by Harris and Smith`" can be generalized into the matcher "`the results by <author> and <author>`". The algorithm for pattern generalization is described in detail and exemplified in (Boisson and Shahmehri, 2000).

Input:
- p: a pattern
- T: a set of templates
- C: a set of concepts
- m: a measure

Begin
$T' = \emptyset$
$t = \texttt{patternGeneralization}(p, C)$
if $\exists t_i \in T$: $\texttt{identify}(m_t,\ m_{t_i})$ then
    $s_{t_i} = s_{t_i} \cup \{t\}$
    $T' = T$
    exit
else
    `similar = false`
    for each $t_i \in T$ do
    if $\texttt{areSimilar}(t, t_i, m)$ then
        `similar = true`
        $t' = \texttt{groupTemplates}(t, t_i)$
        $t'' = \texttt{generalizeTemplate}(t', C)$
        $T' = T' \cup \{t''\}$
    else $T' = T' \cup \{t_i\}$
    end for
    if not `similar` then $T' = T' \cup \{t\}$
End

Figure 5: Algorithm: similarity measures to manage the set of templates

The second step consists of checking if the matcher of the new template can be identified by the matcher of other existing templates. If this is the case, the new template is added to the set of generators of these templates and the algorithm ends.

The third step checks if the new template is similar to any other existing templates. If this is the case, the similar templates become the generators of a new template.

For the instruction $\texttt{areSimilar}(t, t_i, m)$, we have implemented a grammar based version and a word based version of m, the method which measures the similarity between the templates. The results of our experiment, for each version, are given in section 5. Both versions adopt a combination of the syntactic and operation similarities (cf. similarities 2 and 3 from figure 4). Thus the templates that trigger different operations are not similar, and the templates that have the same operation can be syntactically similar with respect to m. We keep both versions of the measures as independent from the application as possible:

- Grammar based version: the rules of the grammar specify the deletion of words which can easily differ from one pattern to another: prepositions and determiners.

- Word based version: we measure the edit distance. For the threshold we use the formula $|\text{length}(m_{t_1})$ - $\text{length}(m_{t_2})| + \frac{min(length(m_{t_1}),length(m_{t_2}))}{3}$ which seems to work well in practice.

The instruction "`groupTemplates(`t`, `t_i`)`" generates a template t' such that its matcher $m_{t'}$ can be used to identify at least all the patterns that t and t_i can identify (i.e. $(P_t \cup P_{t_i}) \subset P_{t'}$) where t and t_i are the generators of t' (i.e. $s_{t'} = \{t, t_i\}$). For example, given the templates t and t_1 with the matchers m_t = "`in the precedent article`" and m_{t_1} = "`the other article`", the instruction `groupTemplates(`t`, `t_1`)` can produce t' such that: $m_{t'}$ = "`(in) the [precedent|other] article`" and $s_{t'} = \{t, t_1\}$.

The instruction "`generalizeTemplate(`t', C`)`" replaces multivalued tokens (e.g. `[precedent|other]`) with a concept to produce the template t'' such that $m_{t''}$ can be used to identify at least all the elements in $P_{t'}$ and and $s_{t''} = \{t'\}$. We provide means to supervise this generalization by defining a threshold for each concept. This threshold is the minimum number of values that a multivalued token should have to qualify its generalization in a given concept. For example: given $m_{t'}$ = "`[the | my | their | an] article`", c_1 = `<determiner>` and threshold$(c_1) = 3$, we can generalize t' into t'' so that $m_{t''}$ = "`<determiner> article`" and $s_{t''} = \{t'\}$.

5 Practical Analysis of the Syntactic Similarity

In the context of the application presented in section 2, the experiment consists of incrementally learning a set of templates from email discussions which contain a number of references to articles. For each email, we perform three actions:

1. Identification of relevant patterns with the set of templates.

2. A user provides information about which patterns were falsely identified and which were missed in the identification.

3. For each new pattern, the algorithm presented in section 4 is used to complete the set of templates.

Our test sets are two email discussions which review scientific articles. The first discussion is composed of 14 mails while the second is composed of 8 mails. The first discussion often uses semantic concepts such as `<author>` and `<conference>`, while the second discussion does more rarely.

The test cases presented in (Boisson and Shahmehri, 2000) show that using the instruction of `patternGeneralization(p, C)` alone has a precision/recall ratio of $\frac{100\%}{21.4\%}$ on the first discussion and of $\frac{57.9\%}{50\%}$ on the second.

The results for our new algorithm, in terms of recall, precision and number of templates are given in table 5. The analysis of the results provides the following information:

- The precision on the second discussion is increased from 57.9% to 100%. This is due to the supervision of the generalization provided by the instruction `generalizeTemplate(`t', C`)`.

- The grammar based measure has no effect on the set of templates for discussion 1, but it performs really well on the set of templates for discussion 2 by reducing the set of templates by 46%.

- On discussion 1, the edit distance finds similarities that the grammar does not. It also produces several templates which have similar generators. We plan to change this behavior.

- On discussion 2, the edit distance generates a set of templates slightly larger than the grammar does.

This experiment shows that our approach is actually able to generate general (recall), and precise templates. It also shows that given a way to handle the similarity of a new template with the sets of generators of the existing templates, our approach provides a smallest set of templates to identify the pattern.

To conclude, grammar based and word based measures capture different syntactical characteristics and thus perform different in different situations. Thus, we propose to use a combination of both methods in two steps:

Step1: Use the grammar method to generate more general templates.

<table>
<tr><td>Discussion 1</td><td>Discussion 2</td></tr>
<tr><td colspan="2">initial patterns</td></tr>
<tr><td>13 patterns
- the results by Hill and Smith
- defined by Hill and Smith
- analyzed by Hill and Smith
- proposed by Thomas
- the proposal by Ferri
- the proposals by Durand and Breton
- adopted by Thomas
- Durand and Breton's semantics
- my book at OxfordUniversityPress
- the IJCAI paper by myself and Jones
- in your article
- anderson describes its reduction to -
- circumscription in an article at ETAI</td><td>11 patterns
- your paper
- in the paper
- the paper
- in another paper by hamilton
- your article
- the article
- this work
- this paper
- my KR-96 paper
- the KR article
- my article</td></tr>
<tr><td colspan="2">template generation with the grammar measure</td></tr>
<tr><td>0 generalized templates, 13 templates
precision/recall = 100%/21.4%
templates:
- the results by Hill and Smith
- defined by Hill and Smith
- analyzed by Hill and Smith
- proposed by Thomas
- the proposal by Ferri
- the proposals by Durand and Breton
- adopted by Thomas
- Durand and Breton's semantics
- my book at OxfordUniversityPress
- the IJCAI paper by myself and Jones
- in your article
- anderson describes its reduction to -
- circumscription in an article at ETAI</td><td>2 generalized templates, 4 templates
precision/recall = 100%/54%
generalized templates:
- [in] <determiner> paper
- <determiner> article
templates:
- in another paper by <author>
- this work
- my <conference> paper
- the <conference> article</td></tr>
<tr><td colspan="2">template generation with the edit distance measure</td></tr>
<tr><td>11 generalized templates, 1 template
precision/recall = 100%/21.4%
generalized templates:
- (defined|the results) by <author> and <author>
- (analyzed|the results) by <author> and <author>
- (proposed|the results) by <author> [and <author>]
- the (proposal|results) by <author> [and <author>]
- the (results|proposals) by <author> and <author>
- (adopted|the results) by <author> [and <author>]
- the (results |<conference> paper) by <author> and <author>
- (proposed|the proposal) by <author>
- <VBN> by <author>
- (my book|<author> described its reduction to circumscription in an article) at (<editor>|<journal>)
- [<author> described its reduction to circumscription] in (your | an) article [at <journal>]
template:
- <author> and <author> 's semantics</td><td>6 generalized templates, 1 template
precision/recall = 100%/54%
generalized templates:
- [in] <determiner> paper
- (your|in another) paper [by <author>]
- your (paper|article)
- <determiner> [<conference>] paper
- the [<conference>] article
- <determiner> article
templates:
- this work</td></tr>
</table>

Table 1: Results

Step2: Use the edit distance to generate more general templates from the list of templates that did not get generalized during the first phase. This second step allows for the generation of a unique template for the patterns "`<determiner> <conference> paper`" and "`the <conference> article`", for example.

6 Related Work

There are several other word based approaches that can be used to measure the syntactical similarity of two sequences of tokens. The Dice coefficient (Frakes and Baeza-Yates, 1992) is an example. We use the edit distance because it is one of the most popular approaches. The command agrep (Wu and Manber, 1992), for example, is based on it.

Many applications based on natural language understanding are prudent when it comes to the usage of a grammar. In (Atwell et al., 1993), the authors state that it is not possible to fully describe an unrestricted Natural Language such as English with a grammar. Moreover building a grammar even for specific applications is not an easy task since one has to think of all the possible cases that could occur. Thus other techniques such as neural networks or probabilities are employed to complete or disambiguate the parsing that can be done with a grammar. Our approach keeps a very simple and general grammar and completes it with a language independent method, the edit distance.

The wrapper language token-templates (Thomas, 2000) allows to build templates that can be used in conjunction with Logic Programs. This language needs a semistructured (i.e. annotated) text as input while we handle non documented texts.

7 Conclusion and Future Work

Our long term goal is to establish automatic support for repetitive applications on texts. As a step towards this goal we look at the automation of the identification of patterns using templates. We proposed to apply syntactic similarity measures to the management of templates. We used two categories of similarity measures: grammar based and word based. We have tested the performance of templates generated using each category of measure on two sets of patterns, in terms of number of templates, precision and recall. As a result we propose to use a combination of both measures since each measure captures different characteristics of the patterns and their performances are complementary. Future work continues to investigate the management of templates by looking at the process of refinement for templates and methods to incrementally adapt the ontology.

Acknowledgements: This research has been supported by the Swedish Research Council for Engineering Sciences (TFR). The author thanks her supervisor, Nahid Shahmehri, for her constant support and valuable comments on her work. The author also thanks Patrick Lambrix and Lena Strömbäck for their helpful comments on earlier versions of this paper.

References

Eric Atwell, Simon Arnfield, George Demetriou, Steve Hanlon, John Hugues, Uwe Jost, Rob Pocock, Clive Souter, and Joerg Ueberla. 1993. Multi-level disambiguation grammar inferred from english corpus, treebank and dictionary. In *Grammatical Inference: theory, applications and alternatives*, pages 91–98.

Ricardo Baeza-Yates and Berthier Ribeiro-Neto, 1999. *Modern Information Retrieval*, chapter 6, page 148. ACM Press Books.

Cécile Boisson and Nahid Shahmehri. 2000. Template generation for identifying text patterns. In *Proceedings of the International Symposium on Methodologies for Intelligent Systems.*

J. Cowie and W. Lehnert. 1996. Information extraction. *Communications of the ACM*, 39(1):80–91, Jan.

ETAI. 1997. Electronic transactions on artificial intelligence. http://www.ida.liu.se/ext/etai/.

William B. Frakes and Ricardo Baeza-Yates, editors, 1992. *Information Retrieval - Stemming Algorithms*, chapter 8. Prentice Hall PTR.

R. Gaizauskas, T. Wakao, K. Humphreys, H. Cunningham, and Y. Wilks. 1995. Description of the lasie system as used for muc-6. In *Proceedings of the 6th Message Understanding Conference (MUC-6)*, pages 207–220.

J. Hobbs, D. Appelt, J. Bear, D. Israel, M. Kameyama, M. Stickel, and M. Tyson. 1996. Fastus: a cascaded finite-state transducer for extracting information from natural-language text. In E. Roche and Y. Schabes, editors, *Finite State Devices for Natural Language Processing.* Cambridge MA: MIT Press.

W. Lehnert, J. McCarthy, S. Soderland, E. Riloff, C. Cardie, J. Peterson, F. Feng, C.Dolan, and

S. Goldman. 1993. Description of the circus system as used for muc-5. In *Proceedings of the 5th Message Understanding Conference (MUC-5)*, pages 277–291.

Pernilla Qvarfordt and Arne Jönsson. 1998. Effects on using speech in timetable information system for the www. In *Proceedings of the International Conference on Speech Language Processing*, pages 1635–1639.

Lena Strömbäck and Arne Jönsson. 1998. Robust interpretation for spoken dialogue systems. In *Proceedings of the International Conference on Speech Language Processing*, pages 491–495.

Bernd Thomas. 2000. Token-templates and logic programs for intelligent web search. *Journal of Intelligent Information Systems*, 14:241–261, May-June.

Sun Wu and Udi Manber. 1992. Fast text searching: allowing errors. *Communications of the ACM*, 35(10):83–91, October.