

Transformation-based Learning in Document Format Processing

James R. Curran and Raymond K. Wong

Basser Department of Computer Science
Madsen Building, F09
University of Sydney
N.S.W. 2006, Australia
{jcurran, wong}@cs.usyd.edu.au

Abstract

Recent work in computational linguistics (Brill 1992; 1994) has described a transformation-based learner with impressive accuracy, speed and a lucid, concise representation. This work presents a set-based formal model of ambiguity, tagging and the transformation-based learning paradigm. We apply the model to the automatic learning of document format generation and recognition on multiple levels of structural semantics. This supports general applicability of the model and results in a novel linear time document format processor.

Introduction

Tags are labels that can represent semantics in a markup language or system. The trend in electronic authoring is to use markup languages that describe *content* rather than *form*. Content tags describe the structural semantics of a document whereas form tags describe document formatting which only implicitly represents structural semantics. For instance, the XML tag <TITLE> describes the title of the document, whereas the HTML tag describes a bold font which may represent the title of the document or a definition or one of many other document structures. Format *generation* involves rendering a content-tagged document with format tags. Format *understanding* involves rendering a format-tagged document with content tags.

The structural semantics represented by formatting are ambiguous and dependent on the context and content of the formatted text. The manner in which these semantics are represented by document formatting is similar to the ambiguous, context dependent representation of semantics in natural language. A Part of Speech (POS) tag describes the role that a word plays in a sentence, such as noun or adjective. A POS tag is context dependent because, although words often have multiple POS tags, not all POS tags apply in every sentence. POS tagging is similar in many aspects to generating or understanding document formatting.

POS tagging has been a popular target for automatic machine learning strategies using large corpora in computational linguistics. Much of this work has

centred around Markov-model (MM) based POS taggers (Jelinek 1985; Church 1988; Cutting *et al.* 1992). A transformation-based POS tagger is presented (Brill 1992; 1994) which achieves results comparable with MM taggers. The tagger learns a simple, lucid set of *transformations* rather than a large table of statistical coefficients. The transformation paradigm has been applied to other linguistic problems including prepositional phrase attachment (Brill & Resnik 1994) and syntactic parsing (Brill 1993). Issues and results in transformation-based learning are discussed in depth in (Brill 1995; Ramshaw & Marcus 1994). A sequence of transformations can be converted into a finite-state transducer (FST) that performs transformations in linear time with the sentence length which is ten times faster than MM tagging (Roche & Schabes 1997).

This work is concerned with presenting a formal model of ambiguity, tagging and transformations using the concept of labelled-set membership. The formal model clarifies the processes of transformation-based learning making it accessible for applications outside POS tagging. Initially, we consider the application of transformations on single sentences and ignore the stochastic aspects of learning transformations. Later, we recognise that transformations are in general applied to a large set of word sentences such as documents or corpora. Transformation-based document format generation and understanding are then described using the model. Further work will involve quantitative analysis of the technique.

Words and Sentences

A word w is an atomic symbol. A vocabulary \mathcal{V}_w is the exhaustive finite set of words used in a system. A sentence \mathbf{W} defined on a vocabulary \mathcal{V}_w is a finite sequence of words in \mathcal{V}_w . A sentence \mathbf{W} can be divided into subsequences called partial sentences. $\mathbf{W}_{x,y}$ is the subsequence containing the words w_x to w_y .

$$\mathbf{W} = \langle w_i \mid w_i \in \mathcal{V}_w \rangle \quad i = 1, \dots, n \quad (1)$$

$$\mathbf{W}_{x,y} = \langle w_x, \dots, w_y \mid 1 \leq x \leq y \leq n \rangle \quad (2)$$

There exists an implicit *start word*, $w_0 = \triangleright \notin \mathcal{V}_w$, and *end word*, $w_{n+1} = \triangleleft \notin \mathcal{V}_w$, of the sentence. These differentiate the beginning and end of the sentence from

the beginning and end of partial sentences derived from that sentence.

There exists a *null word* $\varepsilon \notin \mathcal{V}_w$. Null words can be inserted before the \triangleright word and/or after \triangleleft word without changing the sentence content. Thus $w_{i<0} = \varepsilon$ and $w_{i>n+1} = \varepsilon$.

Word Instances, Semantics and Ambiguity

A word instance \bar{w} is a word with associated syntactic and semantic attributes that is determined by relationships with and between surrounding words (and therefore word instances) in a sentence. Thus word instances only exist as a consequence of sentences, whereas words exist both as monatomic symbols and in sentences. Words are the concrete objects that systems can represent internally and manipulate directly.

Word instances formalise meaning and semantics. A word instance is a particular meaning that a word represents in a particular linguistic context. It is important to realise that word instances can be partitioned into disjoint sets because their associated semantic and syntactic information makes them unique.

A word w represents a word instance \bar{w} in a sentence \mathbf{W} , $w \mapsto \bar{w}$, if the relationships between the words (and therefore word instances) in \mathbf{W} determine the semantic and syntactic attribute values that define \bar{w} .

Multiple distinct instances of a word are necessary to capture the ambiguity resulting from identical words (word forms) that do not have identical syntactic or semantic attributes (word senses or semantics). Some ambiguity can only be resolved outside the scope of the sentence such as by using the communicators' common knowledge.

Properties, Attributes and Descriptions

A property or attribute *value* is a disjoint set of objects which display the same property or attribute value. An object with a particular property value is a member of the disjoint set of objects which display the property value. Objects that display a particular property are partitioned into one of these disjoint sets. If there is an object that appears to violate the disjoint sets constraint, the sets available to partition the object may not be fine-grained enough or the object may be too general and appears to have a property that it does not have a unique value for. The *property* or *attribute* is itself defined by the set of disjoint sets into which objects that are described by that attribute can be partitioned into.

Example 1 An object that is coloured **red** is a member of the set of objects that are **red**. A **stop-sign** and **blood sample** are members of the **red** set. An **apple** is a member of the property set **red** but is also a member of the property set **green**. The disjoint set violation can be resolved by either defining a new property set **red-or-green** which takes the elements that **red** and **green** have in common or by making

the **apple** object more specific such as **red-apple** or **granny-smith-apple**.

A property value *describes*, \uparrow , an object and the object *matches*, \downarrow , a property value if the object is a member of the property value set. Matching and description are inverse relationships¹. Description and matching can be applied to non-disjoint sets in a similar way.

Describing Word Instances and Tags

A tag t is a word that *labels* the set of word instances $\bar{\mathcal{W}}_t$ that define an attribute value. A tag vocabulary \mathcal{V}_t defines a semantic or syntactic attribute by defining what labels a particular tag t can hold. \mathcal{V}_t defines what disjoint sets the word instances can be partitioned into. A tag t *describes* a particular word instance \bar{w} , $t \uparrow \bar{w}$, if and only if \bar{w} is a member of the set of word instances that t labels.

$$t \uparrow \bar{w} \iff \bar{w} \in \bar{\mathcal{W}}_t \iff \bar{w} \downarrow t \quad (3)$$

A tag sentence \mathbf{T} *describes* a word sentence \mathbf{W} , $\mathbf{T} \uparrow \mathbf{W}$, if each tag $t_i \in \mathbf{T}$ describes the corresponding \bar{w}_i represented by $w_i \in \mathbf{W}$.

$$\langle t_1, \dots, t_n \rangle \uparrow \langle w_i \mid w \mapsto \bar{w}_i \wedge t_i \uparrow \bar{w}_i \rangle \quad (4)$$

Example 2 The sentence “**Cows** are female bovine animals.” is tagged as a definition using a font style vocabulary $\mathcal{V}_{STY} = \{N, B, I, \{B, I\}\}$ describing normal, bold, italic and bold-italic. The format tag sentence which represents the sentence is:

$$\mathbf{T}_{STY} = \langle B, N, I, I, I, N \rangle$$

Example 3 Consider a content tagging vocabulary $\mathcal{V}_{CON} = \{DEF, KEY, -, \dots\}$ used to tag the Example 2. The resultant content tag sentence is

$$\mathbf{T}_{CON} = \langle KEY, -, DEF, DEF, DEF, - \rangle$$

Theorem 1 (Multiple Tag Equivalence) A single tag sentence has the same descriptive power as multiple tag sentences.

Proof 1 Consider two tag sentences and their respective vocabularies that describe a word sentence

$$\mathbf{T}_1 \text{ and } \mathcal{V}_1, \mathbf{T}_2 \text{ and } \mathcal{V}_2$$

These tag vocabularies can be disjoint or otherwise. Construct a new tag vocabulary that is the Cartesian product of the two vocabularies:

$$\mathcal{V}_{1 \otimes 2} = \{(t_1, t_2) \mid t_1 \in \mathcal{V}_1, t_2 \in \mathcal{V}_2\}$$

This new tag vocabulary contains all of the information that both \mathcal{V}_1 and \mathcal{V}_2 contain. These new n -tuple complex tags are denoted t^n to indicate the number of dimensions in the tuple. A simple tag has the same expressive power as a complex tag because an n -tuple of tags can be mapped to a single unique tag by concatenation of the tags in the n -tuple. However it is conceptually useful to consider it an n -tuple.

Construct the new tag sentence as follows:

$$\mathbf{T}_{1 \otimes 2} = \langle t_i \mid t_i = (t_{1i}, t_{2i}) \rangle \quad i = 2, \dots, n - 1$$

¹Only description is defined below but whenever a describes b , b matches a in the same manner.

Clearly this new tag sentence expresses all of the information that both \mathbf{T}_1 and \mathbf{T}_2 express. The inductive generalisation from two tag sets to n tag sets using n -tuple tags is clear. \square

The resultant Cartesian product vocabulary is

$$\mathcal{V}_{1 \otimes \dots \otimes n} = \mathcal{V}_1 \otimes \dots \otimes \mathcal{V}_n$$

The cost of this compression into one tag sentence is a much larger vocabulary $|\mathcal{V}_1| \times \dots \times |\mathcal{V}_n|$.

Example 4 A single tag sentence and vocabulary can be generated from Examples 2 and 3. The new vocabulary is constructed using the Cartesian product:

$$\mathcal{V}_{STY \otimes CON} = \{(N, DEF), (N, KEY), (N, -), \dots, (BI, -)\}$$

The new tag sentence uses the new vocabulary to represent the information from both tag sentences: $\mathbf{T}_{STY \otimes CON} = \langle (B, KEY), (N, -), (\{B, I\}, DEF), \dots \rangle$

The definition of simple and complex word descriptor tags as words themselves is powerful. It allows for even more abstract descriptions of the original sentences by defining tags describing the tags ad infinitum. Symmetrically, a tag can represent the symbolic value of the word, that is, a word is a tag that represents itself.

Terms and Contexts

A term is a generalisation of a tag. The term c labels a set of tags, \mathcal{T}_c , in the same way that tags label a set of word instances. A term c can be a tag, $c = t \in \mathcal{V}_t \cup \{\varepsilon, \triangleright, \triangleleft\}$, in which case it labels $\mathcal{T}_c = \{t\}$. Otherwise the term is the special term $c = * \notin \mathcal{V}_t \cup \{\varepsilon\}$ in which case it labels $\mathcal{T}_c = \mathcal{V}_t \cup \{\varepsilon, \triangleright, \triangleleft\}$. A term c describes a tag t , $c \uparrow t$, if and only if t is a member of the set \mathcal{T}_c that c labels.

A context is a term sentence (Brill 1994). A context \mathbf{C} , $|\mathbf{C}| = n$, describes a partial tag sentence $\mathbf{T}_{x, x+n-1}$, denoted by $\mathbf{C} \uparrow \mathbf{T}_{x, x+n-1}$, if each term $c_i \in \mathbf{C}$ describes the corresponding $t_i \in \mathbf{T}_{x, x+n-1}$. This is referred to as \mathbf{C} describes \mathbf{T} at position x , $\mathbf{C} \uparrow_x \mathbf{T}$.

$$\langle c_1, \dots, c_n \rangle \uparrow_x \langle t_1, \dots, t_k \rangle \iff c_i \uparrow t_{x+i-1} \quad (5)$$

A term a subsumes another term b if a describes all of the tags that b describes, that is, if the set of tags that b labels is the subset of the set of tags that a labels. As a result a term subsumes itself. A context $|\mathbf{C}| = n$ can be used to represent all contexts $|\mathbf{C}'| \leq n$ since $*$ can be used to fill in the remaining terms. This is a result of ε tag sentence padding and $*$ matching.

Theorem 1 shows the descriptive equivalence of single and multiple tag sentence descriptions of a word sentence. This can be extended to terms and contexts using the same Cartesian product method where each element of product is a valid simple term. Complex terms label a set of complex tags. A complex term $c^n = (c_1, \dots, c_n)$ describes a complex tag $t^n = (t_1, \dots, t_n)$, $c^n \uparrow t^n$, if and only if each c_i describes the associated t_i .

$$(c_1, \dots, c_n) \uparrow (t_i \mid c_i \uparrow t_i) \quad (6)$$

For succinctness, we use abbreviations for particular complex tags when the meaning is obvious. Consider a term $c^n = (c_1, \dots, c_n)$. c^n is abbreviated to x if $c_i = x$ and $c_j = *$ for $i \neq j$. c^n is abbreviated to x^n if $c_i = x$ for $i = 1, \dots, n$.

Transformations

A transformation is the process by which tags in partial sentences are substituted in a particular context. A transformation is designed or learnt to substitute the correct tag for an incorrect tag in the sentences that it is applied to. It may inadvertently substitute an incorrect tag for a correct tag in other sentences. Learning transformations involves finding the transformations that tag the target sentences as accurately as possible.

A transformation P is a function from one context to another and is defined by the tuple $P = (\mathbf{C}, \mathbf{C}')$. The initial and final contexts, \mathbf{C} and \mathbf{C}' , describe the partial tag sentence before and after the transformation is applied. The initial and final context are both of length n , $|\mathbf{C}| = |\mathbf{C}'| = n$. The tag sentence length $|\mathbf{T}|$ remains constant throughout the transformation. This implies \triangleright , \triangleleft and ε tags cannot be substituted. They must occur in the same positions in the initial and final contexts.

A transformation is *applied* when a partial tag sentence matches the initial context. Application of a transformation involves term-by-term rewriting of the initial context with the final context where the final context term does not subsume the initial context term. The result of applying the first transformation is

$$\mathbf{P}_1(\mathbf{T}) = \langle P_1 \rangle (\mathbf{T}) = P_1(\mathbf{T})$$

The result of applying the first i transformations is

$$\mathbf{P}_i(\mathbf{T}) = P_i(\langle P_1, \dots, P_{i-1} \rangle (\mathbf{T}))$$

Slots and Templates

A slot s is a generalisation of a term. A template \mathbf{S} is slot sentence and therefore a generalisation of a context. The slot s labels a set of terms, \mathcal{C}_s in the same way that a term labels a set of tags. \mathbf{S} describes a transformation $P = (\mathbf{C}, \mathbf{C}')$ in the usual manner.

A slot s_i can either be the value \circ , \bullet or $*$ which identify an *open* slot, *substitution* slot or *closed* slot respectively. An open slot, $s_i = \circ$, is a position where any term values except $*$ can be used to describe a particular term sentence and the initial context term is the same as the final context term. An open slot describes a position of specification in the template. A substitution slot, $s_i = \bullet$ is an open slot which is available for substitution in a transformation thus the initial context term does not have to be the same as the final context term. A closed slot, $s_i = *$ is a position where only the $*$ any term can be tested to describe a particular term sentence. A closed slot describes a position of generalisation in the template.

$$\mathbf{S} \uparrow P \iff \begin{cases} s_i = \circ & \implies \mathbf{C}_i = \mathbf{C}'_i \neq * \\ s_i = \bullet & \implies \mathbf{C}_i \neq \mathbf{C}'_i \\ s_i = * & \implies \mathbf{C}_i = \mathbf{C}'_i = * \end{cases} \quad (7)$$

Templates describe a set of valid contexts and thus reduce the search space by limiting the number of valid contexts. Using templates makes searching transformation space a tractable problem. The consequence of reducing the search space is an equivalent loss of descriptive power. Care must be taken in choosing templates that provide enough descriptive power to cover as many cases as possible.

Learning Transformations

Before transformations can be applied to the tag sentences the tag sentences must be initialised. This initial tag state could be a uniform, default value to indicate that the word is yet to be assigned a valid descriptive tag. Alternatively, the initial state could be descriptively significant such as the most common tag for the described word (Brill 1994). The advantage here is that the fewer transformations are required to achieve the desired accuracy and thus the tagger is faster and smaller. A disadvantage of using a descriptive initial tag state is that other statistical tag information can be lost.

Learning transformations is an optimisation problem where we are trying to optimise the accuracy of the tagger. Optimisation algorithms rely on evaluation of the transformation. A new transformation cannot just be applied to the erroneously tagged sentences because it may inadvertently alter correctly or incorrectly tagged sentences. Statistical evaluation of a transformation is expensive. The optimisation algorithm used in (Brill 1994) is an error-driven greedy algorithm that adds the transformation to the sequence that most improves the tagging accuracy. The evaluation function is the net improvement in accuracy of each transformation.

Learning Document Formatting

A transformation-based system is defined by the mapping to the sentence and word vocabulary model, the tag vocabularies, the initial state and the templates that can be used to learn transformations. In this section we will provide descriptions of each of these elements for a system that can perform either document format generation or understanding.

Input Mapping and Word Vocabularies We will map a whole text document to a single word sentence. This results in a simple description of text whitespace as falling between words rather than between sentences in the model. Each text whitespace in the document is mapped to the special whitespace word \square . Text whitespace between text sentences is mapped to two \square words. A whitespace word is inserted after \triangleright and before \triangleleft which describes whitespace before the start and after the end of the document. Text words and punctuation are mapped to separate words.

Tag Vocabularies The description of the tag vocabulary includes both the content and format tags. Theorem 1 states that each independent content or format

attribute can be described by different tag vocabularies, which can then be combined to describe the complete tag vocabulary. Most markup languages such as HTML and XML use markup with an extent or scope that describes all of the text within that extent or scope. All words within the extent of the markup are tagged with the same tag. When the markup is only applied in mutual exclusion, that is, an extent cannot overlap or subsume another, then the markup tags can form a vocabulary directly. When mutual exclusion does not hold, the power set of the markup tags must be used as the tag vocabulary. However, if a tag combination does not make sense or is equivalent to another tag combination then it can be eliminated.

Example 5 *For instance, if bold and italic font styles are mutually exclusive then the tag vocabulary $\{N, B, I\}$ identifying normal, bold and italic can be used to describe font styles. If bold and italic are not mutually exclusive then the power set $\{N, B, I, \{N, B\}, \{N, I\}, \dots, \{N, B, I\}\}$ is the tag vocabulary. Since normal font attribute has no effect, some equivalent are eliminated reducing the tag vocabulary to $\{N, B, I, \{B, I\}\}$*

Recursive scopes such as sections, subsections and subsubsections present another problem. Simple tag vocabularies cannot represent possibly infinite recursion. However, in most practical documents the recursion is bounded by a small constant. In this case depth numbered tags can be introduced such as SEC1, SEC2, SEC3 and so on.

Here we use two independent content tag vocabularies and sentences. Structural semantics for grouping and hierarchy such as chapter, section, bibliography, abstract or example are described by a single tag sentence and vocabulary. These markup tags are usually recursive and often subsume others. However they normally occur in a hierarchical sequence only. These structures are large and one extent often starts where the previous one finished. Incidental structural semantics such as examples, definitions, keywords or links often occur in overlapping extents such as keywords used in definitions. These structures are smaller and the extents are often isolated. Notice that some structures such as examples could fall into group or incidental semantics depending on the structures of the documents being described. Content tags on whitespace indicate where structures begin and end.

Format tag vocabularies and sentences describe document formatting such as font style, size or typeface and family. Whitespace tags specify whitespace formatting options for the \square word such as newline, double newline, vertical and horizontal tab stops, ruled lines, indentation or null whitespace. Miscellaneous text features such as boxes, numbers or bullets may be described as well. These whitespace and miscellaneous features can be described by a combined tag vocabulary.

The result is a 5-tuple complex tag for Cow in Example 2 would be

(SEC2, KEY, B, 12pt, Times, -)

If the cardinality of the resultant cartesian product vocabulary becomes intractible then some tag vocabularies can be combined or eliminated with an associated loss of descriptive power.

Initial Tagging For document format generation the content tags are known and the form tags are unknown.

(SEC2, KEY, ?, ?, ?, ?)

The initial state of the tagger is the default formatting tags for that particular tag vocabulary. The initial tagging of the input word is

(SEC2, KEY, N, 11pt, Times, -)

For document format understanding the format tags are known and the form tags are unknown.

(?, ?, B, 12pt, Times, -)

The initial state of the tagger is the default content tags which are no group structure and paragraph incidental structure. The initial tagging of the input word is

(-, PAR, B, 12pt, Times, -)

Templates Templates must be able to represent the relationships between content and form tags. Applying format tags to content tags is rarely ambiguous since the unique content tags are represented by one of a limited number of formatting styles.

Format generation predominantly involves close relationships between tags and local relationships within individual complex tags. Close relationships are described by short templates and local relationships are described by unity length templates. Longer templates are only required to describe formatting between structures such as the start and end of sections. In format generating systems the substitution slots are used for format tag positions and the open slots are used for content tag positions within the complex tag.

Example 6 Relationships between the sectional content tags and the miscellaneous format tags are described by the $(\circ, *, *, *, *, *, \bullet)$ template. Relationships between the sectional and incidental content tags with the font style tags are described by the $(\circ, \circ, \bullet, *, *, *)$ template.

Applying content tags to format tags is more ambiguous since the format tags often represent many different content tags in different contexts with overlapping and subsumed extents. Longer templates play a more significant role in format understanding because some ambiguity cannot be resolved with short, local templates. In format understanding systems the substitution slots are used for content tag positions and the open slots are used for format tag positions within the complex tag.

Example 7 Relationships between sectional content tags separated by a word can be described by the $\langle (\circ, *, *, *, *, *), (\circ, *, *, *, *, *), (\circ, *, *, *, *, *) \rangle$ template.

This completes the definition of the document formatting transformation-based system. The error-driven greedy algorithm described in (Brill 1994) is used to learn the document formatting transformations from a corpora containing documents with both content and

form tags. The result of the learning process is a sequence of transformations that can be applied to unseen documents to generate or understand document formatting. Using (Roche & Schabes 1997) this sequence can be converted to a FST in which tags in linear time.

Conclusion

We have contributed a new set-based formal model of transformation-based learning which incorporates ambiguity and tagging. By applying this model to automatic document format processing we have shown that the model is applicable outside the domain of pure computational linguistics. In this process we have developed a solution to document format generation and understanding that inherits the advantages of the successful (Brill 1994) POS tagger. Further work will evaluate the accuracy of the tagger and present templates that best describe the relationships between content and form tags.

References

- Brill, E., and Resnik, P. 1994. A transformation-based approach to prepositional phrase attachment disambiguation. In *Proceedings of the Fifteenth International Conference on Computational Linguistics*.
- Brill, E. 1992. A simple rule-based part of speech tagger. In *Proceedings of the Third Conference on Applied Natural Language Processing*.
- Brill, E. 1993. Automatic grammar induction and parsing free text: A transformation-based approach. In *Proceedings of the 31st Meeting of the Association for Computational Linguistics*.
- Brill, E. 1994. Some advances in transformation-based part of speech tagging. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*.
- Brill, E. 1995. Transformation-based error-driven learning and natural language processing: A case study in part of speech tagging. *Computational Linguistics*.
- Church, K. 1988. A stochastic parts program and noun phrase parser for unrestricted text. In *Proceedings of the Second Conference on Applied Natural Language Processing*.
- Cutting, D.; Kupiec, J.; Pedersen, J.; and Sibun, P. 1992. A practical part-of-speech tagger. In *Proceedings of the Third Conference on Applied Natural Language Processing*.
- Jelinek, F. 1985. Self-organized language modelling for speech recognition. In Skwirzinski, J., ed., *Impact of Processing Techniques on Communication*. Dordrecht.
- Ramshaw, L., and Marcus, M. 1994. Exploring the nature of transformation-based learning. In Klavans, J., and Resnik, P., eds., *The Balancing Act : Combining Symbolic and Statistical Approaches to Language*. The MIT Press.
- Roche, E., and Schabes, Y., eds. 1997. *Finite-State Language Processing*. The MIT Press.