
Production Systems and Information Engineering
Volume 7 (2015), pp. 33–42

ROLE OF WORD REPRESENTATION FORMAT IN MORPHOLOGY LEARNING

László Kovács
University of Miskolc, Hungary
Department of Information Engineering
kovacs@iit.uni-miskolc.hu

Daniel Hládek
Technical University of Kosice, Slovak Republic
Faculty of Electrical Engineering an Informatics
daniel.hladek@tuke.sk

[Received September 2015 and accepted November 2015]

Abstract. This paper provides a survey on the available methods for induction of morphology grammars using statistical methods. The selected word and character representation forms play a crutial role in the efficiency of the generated grammar rule system. The performed analysis covers a wide range of methods where the main focus is set on the decision tree based rule trees which provides an efficient approach for large problems too. Regarding the word representation formats, the paper shows the main benefits of the attribute-based approach.

Keywords: morphology analysis, grammar induction, word representation formats

1. Morphology

The morphology is a linguistic discipline analyzing the internal components of words in expression of syntactical roles. In the case of syntetic languages, the morphemes are the primary tools to determine the meaning of the words. The most widely used implementation forms of morpheme units are the affixes. In concatenative morphology, the different components (morphemes) can be composed into a sequence and there is a clear boundary between the different components. On the other hand, some languages use also non-concatenative morphology rule too. The past tense of irregular verbs in English language is based on this kind of transformation. In the Ngiti [4] language, the plural of a noun is generated by replacing the last two syllables with high tone syllables:

singular: kama, plural: km.

Current morphology analyzer, like the Humor analyser for the Hungarian language [1] uses a lexicon on the possible surface forms. The implemented program performs a search on the input word form for possible analyses. It looks up morphs in the lexicon the surface form of which matches the beginning of the input word (and later the beginning of the yet unanalyzed part of it). The lexicon may contain not only single morphs but also morph sequences. These are ready-made analyses for irregular forms of stems or suffix sequences, which can thus be identified by the analyzer in a single step, which makes its operation more efficient [7, 8].

A transformed word has usually a stem part which relates to the base lexeme word. The inflectional part denotes the modified parts of the word. Considering inflectional part, it can be subcategorized based on the position of the modification: prefix, suffix and infix part. The prefix part occurs at the beginning of the word, suffix is at the end of the word and infix part is within the stem word. The circumfix denotes the case where both prefix and suffix parts are used. Based on the degree of morphology transformations in the words, there are isolating and polysynthetic languages. The isolating languages do not use any morphological transformation, while polysynthetic languages allow considerable number of inflectional morphemes for a lexeme word. Another categorization system is based on the meaning of the affix parts. In case of fusional languages, the same morpheme unit can express many different semantic roles at the same time. For example, the morpheme s in the word walks denotes the following properties: present tense, singular and 3rd person. The other group of languages is called agglomerative languages as they use a unique morpheme for every semantic role. These morpheme units are usually well separable from each other.

Due to the big variety and irregularity of morphology, it is a big challenge in computational linguistic to develop an efficient learning method for induction of morphology rules. In the literature, current tools are primary tailored only on the simple concatenative morphology using no fusional components. The TASR method [6] for example builds up a tree of suffix rules based on the training samples. The method provides efficient execution cost characteristics but it has been proven inefficient in generalization of the rules for untrained data.

2. Learning methods

2.1. TASR - a decision tree method

A widely used tree-based method is the TASR method [6] which is based on a tree of suffix rules. A suffix rule is defined with a rule $LHS \rightarrow RHS$, where LHS is the left-hand suffix and RHS is the right-hand suffix. This means that LHS from the input will be transformed to RHS in the output. For example, the training word pair alma → almt indicates the following suffix rules:

- alma → almt
- lma → lmt
- ma → mt
- a → t

A dependency can be observed among these rules based on the alignment relationship. The $LHS \rightarrow RHS$ suffix rule is aligned with the $LHS' \rightarrow RHS'$ suffix rule iff $LW = con(s, LHS) = con(s', LHS')$ and $RW = con(s, RHS) = con(s', RHS')$ where symbol con means string concatenation and s, s' are prefixes and LW, RW are arbitrary words.

The child suffix rule $LHS \rightarrow RHS$ is subsumed by the parent suffix rule $LHS' \rightarrow RHS'$ iff $\exists x \in \Sigma : (LHS = con\,(x, LHS') \ \wedge RHS = con\,(x, RHS'))$.

The suffix rule tree based on the alignment relationship of the left hand suffixes, meets the following properties:

- The root's $|LHS|$ is minimal.
- Each node contains the rule with the highest frequency, where that rule is not subsumed by the parent's rule.
- For every child node, $LHS' = con\,(x, LHS)$ where $x \in \Sigma$ and LHS is the parent rule.

The tree building starts with an empty root node, by extracting all the aligned suffix rules according to the definitions above. For every node we have to find the rule with the highest frequency and check whether or not it is subsumed by the parent rule. In the latter case the found rule is the winning rule and we can append the new node to the tree.

After this recursive building algorithm, we can start using the tree, this time in a bottom-up fashion. We traverse the tree for each input word, and try to find the first matching rule from the deepest level. After we found it,

we can apply the transformation on the input word and return the result.

Of course this method depends heavily on the input training set, and we can only use TASR in case of special languages, because only the suffixes are considered by the algorithm. However, if the language meets the requirement of TASR, it is by far the fastest method.

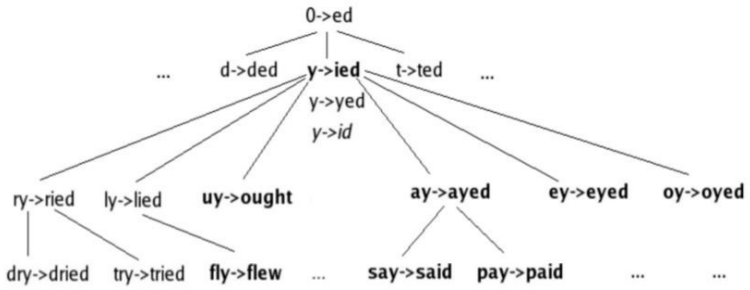

Figure 1. Sample figure (source:[6])

2.2. Ostia - FST method

Another but slightly more complex morphological method is OSTIA or the Onward Subsequential Transducer Inference Algorithm [5] with which belongs to the family of FST (finite state transducer) methods.

A transducer is defined as $t \subseteq \Sigma^* \times \Gamma^*$, $t = \{(s_1, s_2) \mid s_1 \in \Sigma^*, s_2 \in \Gamma^*\}$ where s_1 is the input word and s_2 is the output word.

A rational transducer is a tuple $T = (Q, \Sigma, \Gamma, q_\lambda, E)$ where

- Q is the set of states
- Σ and Γ are the input and output alphabets, respectively
- q_λ is the start state
- $E \subseteq Q \times \Sigma^* \times \Gamma^* \times Q$ is the set of transitions

The sequential transducer is a rational transducer such that $E \subset Q \times \Sigma^* \times \Gamma^* \times Q$ and $\forall\,(q, a, u, q')\,, (q, a, v, q'') \in E \rightarrow u = v \wedge q' = q''$

The subsequential transducer is defined as $T = (Q, \Sigma, \Gamma, q_\lambda, E, \sigma)$ such that the first five components make a sequential transducer and $sigma : Q \rightarrow \Gamma^*$ is the state output function.

The onward transducer meets the following condition

$$\forall q \in Q, a \in \Sigma : lcp\left(\left\{u \mid \left(q, a, u, q'\right) \in E\right\} \cup \left\{\sigma\left(q\right)\right\}\right) = \lambda,$$

where *lcp* returns the longest common prefix of the given word set.

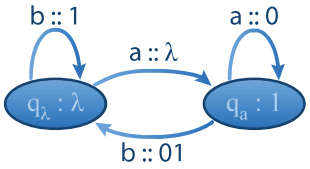

Figure 2. A sample onward subsequential transducer

During production mode, we start from q_λ (start state) and for every input character we change our state and append the output characters on the edge to the output word. After we reach the end of the input word, the end state's output characters are also appended and thus the output word is produced and the algorithm is finished.

There are many applications of the FSA methods in the literature. It can be used among others for learning of word formation too. In the article [2], the formation of noun + noun compounds by computational morphological means is investigated by FST in order to understand how this process should be formalised, modelled and subsequently implemented. In [3], Allomorfessor model is presented using a probabilistic context-free grammar (PCFG). The model extends the morphological segmentation method, Morfes- sor Baseline, to model allomorphy. The proposed unsupervised method discovers common base forms for allomorphs from an unannotated corpus.

2.3. Markov models

In the statistical approaches of NLP problems the Markov model is a dominating method. The Markov processes are based on the assumption that the state value at given time point, depends only on the previous states. In the case of 1-MM, the following assumption is used for every t index:

$$P(s(t) = x|s(t-1)s(t-2)...s(1)) = P(s(t) = x|s(t-1))$$

The main parameters of the Markov model are the

- the probabilities of the initial states
- the probabilities of the state transition

After learning the probability parameters, the probability of the different state sequences can be calculated.

The base Markov model [10] is suitable for problems where the state values can be observed directly. In many problems of morphology and grammars, this condition is not met, as we cann't observe the state, only some related labels are visible. In the case of Hidden Markov Models (HMM), the following components are included into the problem description:

- set of hidden states
- state transition probabilities
- initial state probabilities
- set of output symbol
- symbol transmission probabilities for the states

The goal of the Viterbi algorithm is to determine the hidden state sequence for a given symbol sequence. The algorithm uses a dynamic programming approach, i.e. the global optimum is calculated via sequence of local optimum states. The algorithm determines the most likely state transition path for the given observation sequence O and probability parameters M:

$$argmax_X P(X|O, M)$$

3. Character Representation Formats

In the morphological analysis of the words, the representation form plays a crucial role. The morphology rules are usually based on latent physical and mental parameters and the written form of the words is a result of several transformation steps. The process of morpheme analysis is used in many different application areas having very different process models. In the different environments, different transformation processes can be observed. For example, in NLP interface, the voice to text transformation has a central role, while in a word spelling application, the position of keyboard buttons is an important factor.

3.1. Alphabet representation

This base model is the most widely used model where the alphabetical symbols are applied as building blocks. Most of the proposals like [9] use this approach. This model is the most general one, and it can be used for every application areas. This model is based on the surface level, the processes are working only with the visible symbols. As there are many complex dependencies in the background, the work at the surface level is considered as a more complex problem, it is harder to detect the dependencies. The following table summarizes the benefits and the drawbacks of this model.

Table 1. Alphabet representation

Benefits	**Drawbacks**
- simple representation	- complex relationships
- general	- not very efficient
- many application areas	
- many supporting methods	

3.2. Metric space approach

In a spell checking application, the core measure is the similarity between the words. There exists a lexicon of valid words and the words found in the text should be tested for validity. If the word is not found, the best matching words from the lexicon should be suggested for corrections [11].

The metric space is based on a distance function meeting the following conditions

$$d(a,b) \geq 0,$$
$$d(a,b) = d(b,a),$$
$$d(a,c) \leq d(a,b) + d(b,c).$$

In this problem domain, the word distance metric is used to measure the similarity between the words. The most usual approach is the edit distance, where a dynamic programming algorithm is used to calculate the minimum transformation cost. The most widely used approach is the application of unit costs, i.e unchanged

- costinsert = 1
- costdelet = 1
- costreplace = 1
- costunchanged=0

In the case of special application, like spell check of text written on keyboard, a more specific distance model can provide a better solution. For example, in the case of keyboard, the distance matrix of the letters is based on the position of the letter keys. Using this mode, the implemented spell checker preferences the lexicon words having characters in the neighboring positions.

Table 2. Metric space representation

Benefits	**Drawbacks**
- simple representation	- complex relationships
- better accuracy for special domains	- not very efficient
- many supporting methods	- domain specific distances

3.3. Vector representation

For description of the label parameters, the label should be considered as a compound object. This model enables a deeper investigation and a more accurate representation. For example, in the case of NLP applications, the letter is a representations of the tones and the tones have special physical parameters. Thus the letter symbols can be described with the help of these physical symbols. This representation can improve the efficiency of the rule induction processes as it takes also some of the latent parameters behind the visible letters. It can be assumed namely, that there exists some relationship between the physical parameters of the tunes in the word inflection system.

Using a vector space model, the feature vectors are used to represent the letters where the dimensions correspond to the attributes. The main benefit of the vector model, that the usual calculation methods like Euclidean distance or cosine similarity can be used on a standard way. On the other hand, the vector model is a relatively rigid model, as the dimensionality is fixed. That restriction makes some reasoning operations like generalization more difficult to implement.

Table 3. Vector representation

Benefits	**Drawbacks**
- more accurate representation	- complex relationships
- standard model	- higher calculation costs
- more accurate inference	- rigid representation
- many application areas	
- many supporting methods	

3.4. Attribute set representations

The attribute set representation has a similar goal as the feature vector representation but here, the attributes are considered as elements of a set instead of dimensions. This model also enables a deeper investigation and a more accurate representation. This representation can improve the efficiency of the rule induction processes as it takes also some of the latent parameters behind the visible letters.

This approach is based on an object representation widely used in many information models, namely on the application of a relation context. The main benefit of the set based approach that it enables the application of the standard methods in conceptual modelling. For example, in the FCA (Formal Concept Analysis) model where the objects are described with attribute sets,

the generalization can be used on a natural way. During the generalization, the intersection of the corresponding attribute sets yields the description of the generalized concept. Using sets instead of vectors can extend the set of operations in the rule induction methods.

Table 4. Attribute set representations

Benefits	**Drawbacks**
- more accurate representation - standard model - more accurate inference - many application areas - many supporting methods - flexible representation	- complex relationships - higher calculation costs

3.5. Syllable representation

In some proposal on morpheme analysis, a model best matching the speech format is used to represents the words. The syllable is an atom of the speech and this unit can be used in word analysis too. As the syllable units are closer to the physical level than the letter description, it can provide a more accurate analysis of the hidden relationships. To use a syllable based description, the words should be transformed first into a syllable format. This step is on the other hand, not a trivial task.

Table 5. Syllable representations

Benefits	**Drawbacks**
- more accurate inference - direct representation	- complex relationships - higher calculation costs - domain specific - few supporting methods

4. Conclusion

In this paper, a survey on the main statistical methods for morphology rule induction was presented with especial highlight on the role of word representation formats. From the viewpoint of generality and cost efficiency, the methods using decision tree structure and an attribute set representation format provide the best option for implementation.

Acknowledgements

The presented research work was partially supported by the grant TÁMOP-4.2.2.B-15/1/KONV-2015-0003. The presented research work was partially supported by the grant TÉT_12_SK-1-2013-0019.

REFERENCES

[1] Prószéky, G. and Novák, A.: *Computational Morphologies for Small Uralic Languages, Inquiries into Words*, Constraints and Contexts. 2005, pp.116-124

[2] Pretorius, L., Viljoen, B., Pretorius, R. and Berg, A.: *Towards a computational morphological analysis of Setswana compounds*, Literator 29(1), 2008, pp.1-20

[3] Kohonen, O., Virpioja, S. and Klami, M.: *Allomorfessor: Towards Unsupervised Morpheme Analysis*, Proc. of Evaluating Systems for Multilingual and Multimodal Information Access, 2009, pp. 975-982

[4] Booij, G. : *Construction Morphology*, Oxford press, 2010

[5] Higuera, C.: *Grammatical Inference Learning Automata and Grammars*, Cambridge University Press, 2010

[6] Shalonova, K., Flach, P.: *Morphology Learning Using Tree of Aligned Suffix Rules*, ICML Workshop on Grammatical Inference, 24th International Conference on Machine Learning, 2007.

[7] Akane, T., Mitsuyuki, S. and Akiko, O,: *Development of a Dictionary for the Morpheme Analysis of Agricultural Information*, Proceedings of WCCA 2008, pp. 449-455.

[8] Frakes, W. and Fox, C.: *Strength and Similarity of Affix Removal Stemming Algorithms*, ACM SIGIR Forum, 37(1), 2003, pp. 26-30.

[9] Tordai, A. and Rijke, M.: *Four Stemmers and a Funeral: Stemming in Hungarian at CLEF*, Accessing Multilingual Information Repositories, NLCS 4022, 2006, 179-186.

[10] Manning, C. and Schtze, H.: *Foundations of Statistical Natural Language Processing*, MIT Press, 2003

[11] Kovacs, L.: *Efficient dictionary matching of character streams*, Proc. of SISY 2010, 2010, pp. 103-107