Int J Speech Technol (2016) 19:647–654
DOI 10.1007/s10772-016-9356-2

Corpus based part-of-speech tagging

Chengyao Lv[1] · Huihua Liu[1] · Yuanxing Dong[1] · Yunliang Chen[1,2]

Received: 10 April 2016 / Accepted: 21 July 2016 / Published online: 1 August 2016

Abstract In natural language processing, a crucial subsystem in a wide range of applications is a part-of-speech (POS) tagger, which labels (or classifies) unannotated words of natural language with POS labels corresponding to categories such as noun, verb or adjective. Mainstream approaches are generally corpus-based: a POS tagger learns from a corpus of pre-annotated data how to correctly tag unlabeled data. Presented here is a brief state-of-the-art account on POS tagging. POS tagging approaches make use of labeled corpus to train computational trained models. Several typical models of three kings of tagging are introduced in this article: rule-based tagging, statistical approaches and evolution algorithms. The advantages and the pitfalls of each typical tagging are discussed and analyzed. Some rule-based and stochastic methods have been successfully achieved accuracies of 93–96 %, while that of some evolution algorithms are about 96–97 %.

Keywords Natural language processing · POS tagging · Hidden markov models · Support vector machine · Neural networks · Gene expression programming

✉ Yunliang Chen
cyl_king@hotmail.com

Chengyao Lv
yaoyaobox@gmail.com

Huihua Liu
huihua@cug.edu.cn

Yuanxing Dong
dongyx@cug.edu.cn

[1] School of Foreign Language, China University of Geosciences, Wuhan 430074, China

[2] School of Computer Science, China University of Geosciences, Wuhan 430074, China

1 Introduction

Part-of-speech tagging is a basic problem in natural language processing (NLP). Potential applications for part-of-speech (POS) tagging exist in many areas including speech recognition, speech synthesis, machine translation and information retrieval (Greene and Rubin 1971; Varile and Zampolli 1997; Voutilainen 2003; Karkaletsis et al. 2015). POS tagging tries to tag (or label) each word in a sentence with the correct POS.

In POS tagging, each word or punctuation mark in the text is assigned with its tag. Different tagging systems can use different sets of tags. Typically a tag describes a word class and some word class specific features, such as number and gender.

Most POS tagger involves two problems:

(1) Finding the exact tags for each word. This can be easy if the word is in a word tag lexicon, but if the word is unknown, this may be tough to do.
(2) Choosing between the possible tags. This is called syntactic disambiguation, and it has to be solved for each word that is ambiguous in its POS.

Ambiguous words are very common in most languages, for example the English word set "can" be either a noun, an adjective, or a verb.

A lot of work and research has been going on in this area with great success (Jamatia et al. 2015). Most previous works applied different kinds of machine learning algorithms to POS tagging. Two factors that determine the tag of a word are its lexical probability and its contextual probability (Voutilainen 2003; Sun et al. 2008). Some approaches have been adopted, in most reports which can mainly fall into rule-based approaches and statistical approaches (Greene and Rubin 1971; Sun et al. 2008).

Rule-based approaches apply language rules to improve the accuracy of tagging (Brill 1992). However, the monumental manual work required and the need to have qualified people with a working knowledge of linguistics make this approach too inefficient to be practical.

Another rule-based approach "Transformation-Based method" was introduced in Ref. (Brill 1995). Brill pioneered a rule-based tagging approach using the transformation-based learning (TBL) methodology where the rules are not manually constructed, but learned from corpora. The three-phased process starts with an annotator to create the initial state by assigning a tag to each word in the corpus.

Statistical tagging uses large amount of data to establish the language of each situation and neither require knowledge of the rules of the language nor try to deduce them. Most of these systems are based on Decision Tree (Magerman 1995), Hidden Markov model (HMM) (Rabiner 1989; Carlberger and Kann 1999; Thede and Harper 1999; Lee et al. 2000; Brants 2000), Maximum Entropy Model (Ratnaparkhi 1996) or Support Vector Machines (SVM) (Giménez and Marquez 2004).

Using intelligent computing is another choice. A third and rather new approach is tagging with neural networks (Lippmann 1989; Schmid 1994; Marques and Lopes 2001), genetic algorithms (Araujo 2002) and gene expression programming (Lv et al. 2010). In the area of speech recognition neural networks have been used for a decade now. They have shown performances comparable to that of HMM systems or even better (Lippmann 1989; Schmid 1994). Evolutionary algorithms are among the most efficient methods to deal with complex optimization problems (Goldberg 1989). They have been applied to different issues of natural language processing such as query translation (Davis and Dunning 1995), inference of context free grammar (Smith and Witten 1995) and parsing (Araujo 2001; Bernd Bohnet and Joakim Nivre 2012). The POS tagging model with GA is the use of an evolutionary algorithm to find the tagging of new sentences and can achieve an accuracy of 96 % (Araujo 2002).

Gene Expression Programming (GEP) is a revolutionary member in the family of intelligence computing introduced by Candida in 2001 (Ferreira 2001). GEP has achieved a great progress in dressing the problem of machine learning and unknown things prediction. For the first time lv applied GEP to POS tagging, and obtained a high accuracy of 97.4 % (Lv et al. 2010).

With these methods, efficient and fast taggers have been developed. The best reported taggers have attained a high accuracy of 96–98 %. However, although the percentage is high, it is not so good.

Taking these figures into account one may think that POS tagging is a solved and closed problem being this accuracy perfectly acceptable for most NLP systems. So why waste time in finding a new tagger with a higher accuracy? What does an increasing of 0.3 % in accuracy really mean? We think that there is an important reason for thinking that there is still work to do in the field of automatic POS tagging:

A corpus may have thousands of sentences, and each sentence may have an average of around 30 words. The rate of 97 % means that there is one word tagged in error per sentence, 100 in a 100-sentence document, and 1 million in a corpus with 10,000 documents. Since the POS tagging is one of the earlier steps in many natural language processes. Some NLP tasks are very sensitive to POS disambiguation errors which can be found in the domain of Word Sense Disambiguation (Wilks and Stevenson 2000) and Information Retrieval (Krovetz 1997). Starting with an error in each sentence could be a severe drawback, especially considering that the propagation of this error could grow more than linearly.

There are two major problems which keeping tagging accuracy from 100 %: ambiguous words and unknown words. The ambiguity problem stems from the fact that the English word can be a noun, a finite verb or an infinitive. For example, consider this sentence: 'We can can the can' (Voutilainen 2003). The same word can is used in three different syntactic ways: as an auxiliary, a verb, and a noun. For a machine to determine what tag goes with which can is a difficult problem. It is not a difficult problem for a human. When knowing the word's context and the syntax of the sentence, disambiguation ceases to be a problem (Martinez 2011).

The other obstacle in achieving 100 % accuracy in POS tagging is the set of words the tagger had not encountered in its training corpus. This is known as the unknown words problem. The accuracy rates of the methods mentioned above are limited to no more than 95 % while dealing with unknown words. Syntactic parsers, whose dependence on the output of the tagger is critical, will be stumped when encountering a word without a tag. Even with these limitations, taggers are being used in information retrieval (IR), question answering systems, partial parsing, lexical acquisition, information extraction (IE), and text data mining (Manning et al. 1999).

2 Tagging methods

This section presents the main approaches which are frequently used for POS-tagging. Of particular relevance are the exact features which are used by each tagger for the disambiguation of ambiguous and unknown words, so these will be treated in some detail. In Sect 2.1, a rule-based approaches "Transformation-Based method" was introduced and the performance of TBL model was reported.

HMM and SVM model were analyzed in Sects 2.2 and 2.3 which represent the stochastic method. After that, two typical evolution algorithms model: Neural Network and Genetic Expression Programming model were spread out and discussed in subsection D and F.

2.1 POS tagging with transformation-based learning

Transformation-based learning tagging is a typical rule-based tagging where the rules are not manually constructed, but learned from corpora. The TBL paradigm as applied to POS tagging was first described in (Brill 1995). The TBL method is a machine learning technique which takes a tagged corpus as input from which it can learn how to correctly tag a test sample. The tagger learns a set of rules for assigning tags from the training data which gives the least possible errors and applies them to test data.

The algorithm relies on the fact that for the task it is trivial to devise a very simple tagging mechanism which achieves quite reasonable results. Such a method can be used to create an initial annotation of the text. Of course while such an annotation will have a large percentage of tags correct, there will still be a substantial proportion which is incorrectly tagged. The TBL algorithm aims to correct these errors by successively applying rules which correct such errors based on contextual and word-form-derived information.

The learning phase follows with the use of a set of predefined rule (transformation) templates. By instantiating each template with data from the annotated corpus, a set of rules is created. Each rule is then applied to the corpus that has been tagged by the annotator. The output is compared to the manually annotated corpus, which is considered to be the 'truth'. In this step, transformations are learned and listed. These transformations are applied one at a time to the output of the annotator and again compared to the truth. In essence, a single transformation is learned when the 'learner' tries every possible transformation, while keeping a tally of the number of tagging errors after each transformation is tried. The best performing transformation or rule (i.e., the one that 'resulted in the greatest error reduction') is selected. The learning phase ends when there are no transformations that would reduce the error beyond some predefined threshold.

This approach produces results very close to those which use far more mathematically complex algorithms. Brill reports an overall accuracy of 96.6 % on the Penn Treebank WSJ corpus using 900,000 words of training data (split as 600,000 for learning contextual rules and 350,000 for learning rules for unknown words). The only drawback of this scheme is the long training time required, since in each iteration the counts for each possible rule must be regenerated, as previous rule applications will have probably changed the score of that rule since counts were last generated.

One of the most successful approaches to deal with this problem was that devised by Ngai and Florian (2001), which vastly reduces the training time with no reductions in accuracy. The basic idea is to avoid repetition by generating and storing good and bad counts for each rule "r" once at the beginning, and updating the counts only if they are modified by the application of another rule.

2.2 Hidden markov model

A HMM (Dan Garrette and Jason Baldridge 2013; Owoputi et al. 2013) is a typical statistical tagging model that can be used to solve classification problems that have an inherent state sequence representation. The model can be visualized as an interlocking set of states. These states are connected by a set of transition probabilities, which indicate the probability of traveling between two given states. A process begins in some state, then at discrete time intervals, the process "moves" to a new state as dictated by the transition probabilities. In an HMM, the exact sequence of states that the process generates is unknown (i.e., hidden). As the process enters each state, one of a set of output symbols is emitted by the process. Exactly which symbol is emitted is determined by a probability distribution that is specific to each state. The output of the HMM is a sequence of output symbols.

A complete and excellent description of the equations used in the standard Markov model for part-of-speech tagging is found in (Charniak et al. 1993). A text of n words is seen as a sequence of random variables $W_{1\ldots n} = W_1 W_2 \ldots W_n$, and the corresponding tagging is also a sequence of random variables $T_{1\ldots n} = T_1 T_2 \ldots T_n$. A particular sequence of values of $W_{1\ldots n}$ $(T_{1\ldots n})$ is denoted $w_{1\ldots n}$ $(t_{1\ldots n})$. The definition of the tagging problem is then:

$$f(w_{1\ldots n}) = \arg\max_{t_{1\ldots n}} P(t_{1\ldots n}|w_{1\ldots n}) \tag{1}$$

where the operator "argmax" computes the tagging maximizing the probability, according to the model, that word sequence $w_{1\ldots n}$ is tagged $t_{1\ldots n}$. Making the two assumptions (Carlberger and Kann 1999):

$$P(w_i|t_{1\ldots i}, w_{1\ldots i-1}) = P(w_i|t_i) \tag{2}$$

$$P(t_i|t_{1\ldots i-1}, w_{1\ldots i-1}) = P(t_i|t_{1\ldots i-1}) \tag{3}$$

That is, the word itself only depends on its tag, and the tag only depends on the i−1 preceding tags in the text. The tagging problem can now be formulated as formulary 4.

$$f(w_{1\ldots n}) = \arg\max_{t_{1\ldots n}} \sum_{i=1}^{n} P(t_i|t_{1\ldots i-1}) P(w_i|t_i) \tag{4}$$

An unattractive feature of this formulation is that the quantities $P(w_i|t_i)$ are very small and difficult to estimate. Since the reversed conditional probabilities $P(t_i|w_i)$ are much more attractive in this respect, the following is a plausible alternative:

$$f(w_{1\ldots n}) = \underset{t_{1\ldots n}}{\arg\max} \sum_{i=1}^{n} P(t_i|t_{1\ldots i-1})P(t_i|w_i) \quad (5)$$

Both of these equations (and in particular, their corresponding first order Markov model equations) have been used in different stochastic taggers, but in Charniak et al. (1993), the two equations were compared, and Eq. (1) was found to be significantly better when tagging texts with quite a large training text. If all the probabilities are known, the optimal solution to the tagging problem using Eq. (1) is most efficiently computed with dynamic programming using the so called *Viterbi algorithm* (Viterbi 1967). This algorithm avoids the polynomial expansion of a breadth first search by trimming the search tree at each level.

In recent years, different kinds of Markov model had been adopted in POS tagging. In which two methods are worth mentioned.

Merialdo (1994) conducted several experiments which are the use of untagged text in the training of a simple triclass Markov model. Two approaches in particular are compared and combined: using text that has been tagged by hand and computing relative frequency counts, using text without tags and training the model as a hidden Markov process, according to a Maximum Likelihood principle. Training with the tagged corpus used Random Field (RF), and training with the untagged corpus used maximum likelihood (ML). Both approaches are used to determine the probability of a sequence of tags. The ML was done using the forward–backward (FB) algorithm. In the first experiment using tagged text, The RF was used for training and the Viterbi algorithm for tagging. Using 42,000 tagged sentences (approximately 1 million words) an accuracy of 97 % was reported. In the second experiment, untagged text was used (40,000 sentences). The training was done using ML and the tagging using the Viterbi algorithm. The reported accuracy was around 88.4 %.

The second POS tagger is the Trigrams 'n' Tags (TnT) created by (Brants 2000). The tagger makes use of a combination of smoothing using context-independent linear interpolation and word features like suffixes and capitalization. He reports 96.6 % accuracy with an added 0.5 % (97.1 %) accuracy when the model is tested and trained with data from one annotator; that is, data tagged manually by one and the same person (Brants 2000). Brants makes a point of stating that his tagger does as well as maximum entropy taggers, but faster.

A few other POS taggers based on HMMs have been proposed in the last 15 years (Rabiner 1989; Carlberger and Kann 1999; Thede and Harper 1999; Lee et al. 2000). Although none of them have achieved 100 % accuracy, several kinds of taggers using HMMs are wildly used in POS tagging (Yuan Tian and David Lo 2015).

2.3 Support vector machine-based POS tagging

Support vector machines (SVMs) were first applied to POS tagging in Nakagawa et al. (2001). An SVM is a binary classification algorithm based on a geometric interpretation of the feature values for each instance. As detailed in (Cristianini and Shawe-Taylor 2000), given a set of training instances each consisting of a vector of binary or numeric feature values and a true classification $y \, \varepsilon \, \{-1, 1\}$, an SVM learns a classification function $f(x)$ which can be used to classify a test instance with feature vector x. In binary classification problems, the classification rule is then *sgn* ($f(x)$). The classification rule $f(x)$ is dependent on what is known as the *kernel function*, which effectively maps the data into a higher dimensional feature space allowing the correct classification of instance which have non-linearly separable feature values in the original feature space.

Another SVM-based tagger extend binary support vector machines to cover multiclass classification using a strategy known as one-per-class binarisation (Giménez and Marquez 2004). A SVM is constructed for each POS which contains ambiguous lexical items (reportedly 34 for the Penn Treebank), and in the classification stage, the most confident prediction from all of the SVMs is selected as the tag for the word.

The contextual features used in Gimḿenez and M'arquez's tagger include unigrams, bigrams and trigrams of words and POS, derived from the tokens appearing in a context window of two tokens on either side of the target. POS features for ambiguous words which have not yet been tagged can be replaced with ambiguity classes. These nominal features are binarised to act as input to the SVM in the usual way: a nominal feature with k possible values is represented by k binary features each of which is true when the original feature takes one particular value and false otherwise. The accuracy reported for SVMTool is 97.16 % for all tokens and 89.01 % for unknown tokens.

2.4 POS tagging with neural networks

In the area of speech recognition neural networks have been used for a decade now. They have shown performances comparable to that of HMM systems or even better (Lippmann 1989). POS prediction is another area, closer to POS tagging, where neural networks have been applied successfully. Nakamura et al. (1990) trained a 4-layer feed-forward network with up to three preceding POS tags, as input to predict the word category of the next word. The

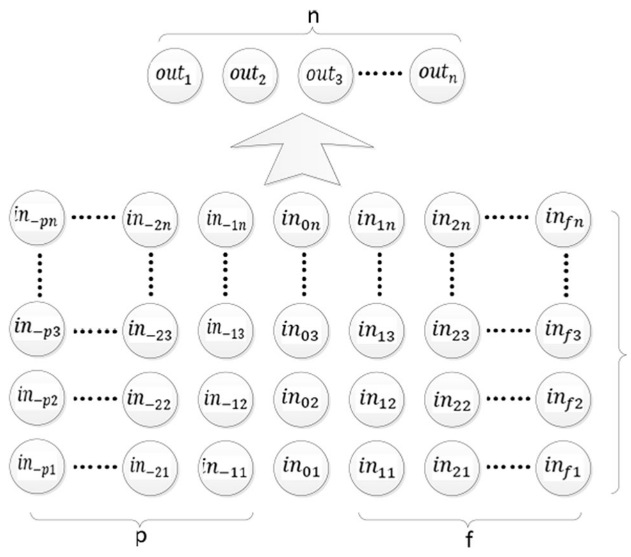

Fig. 1 The structure of the net-tagger

prediction accuracy was similar to that of a trigram based predictor. Using tile predictor, Nakamura's tagger is able to improve the recognition rate of their speech recognition system from 81.0–86.9 %.

Figure 1 shows the structure of the Net-Tagger without hidden layer; the arrow symbolizes the connections between the layers.

In the output layer of the multilayer perceptron network (MLP), each unit corresponds to one of the tags in the tagset. The network learns during the training to activate that output unit which represents the correct tag and to deactivate all other output units in the trained network, the output unit with the highest activation indicates which tag should be attached to the word that is currently processed.

The input of the network comprises all the information which the system has about the parts of speech of the current word, the preceding words and the following words. More precisely, for each POS tag pos_j and each of the $p + 1 + f$ words in the context, there is an input unit whose activation in_{ij} represents the probability that $word_i$ has part of speech pos_j.

For the word which is being tagged and the following words, the lexical POS probability $p(pos_j|word_i)$ is all we know about the part of speech. This probability does not take into account any contextual influences. So, we get the following input representation for the currently tagged word and the following words:

$$in_{ij} = p\left(pos_j|word_i\right),\ if\ i \geq 0 \tag{6}$$

For tile preceding words, there is more information available, because they have already been tagged. The activation values of the output units at the time of processing are here used instead of the lexieal POS probabilities:

$$in_{ij}(t) = out_j(t + i),\ if\ i < 0 \tag{7}$$

Table 1 The comparison of taggers

Method	Accuracy (%)
Net-tagger (Goldberg 1989)	96.22
Trigram tagger (Kempe 1993)	96.06
HMM tagger(Cutting et al. 1992)	94.24

The Net-Tagger (Schmid 1994) was trained on a 2 million word subpart of the Penn-Treebank corpus. Its performance was tested on a 100,000 word subpart which was not part of the training corpus. The training of the tagger took one day on a Sparcl0 workstation and the tagging of 100,000 words took 12 min on the same machine.

In Table 1, the accuracy rate of the Net-Tagger is compared to that of a trigram based tagger (Kempe 1993) and a HMM tagger (Cutting et al. 1992) which were trained and tested on the same data.

The accuracy of Net-Tagger can achieve a little higher than that of the trigram tagger and HMM tagger. The robustness on small training corpora is as good as that of the HMM tagger. Thus, the Net-Tagger combines advantages of both of these methods. The Net-Tagger has the additional advantage that problematic decisions between tags are easy to detect, so that in some cases an additional tag can be given in the output. In this way, the final decision can be delayed to a later processing stage, e.g. a parser. A disadvantage of the presented method may be its lower processing speed compared to statistical methods.

2.5 POS tagging with GEP

In Lv et al. (2010), Lv introduce a rather new evolutionary algorithm, GEP, for POS tagging. GEP is similar to GAs (Araujo 2002) and it differs from GAs mainly in chromosome encoding. GEP encodes individuals as chromosomes and implement them as liner stings with fixed lengths (Ferreira 2001; Zuo et al. 2002). The separation of genotype and phenotype has endowed GEP with more flexibility and power of exploring the entire search space. The chromosomes of GEP are simple and linear. It can be operated by the genetic process easily, and it has the capability to handle complex problems (Ferreira 2003; Zhou et al. 2003; Zuo et al. 2004; Jing et al. 2005; Karakasis and Stafylopatis 2008). GEP has been applied in the problem of machine learning and unknown things prediction and has achieve a great progress (Ferreira 2003; Zhou et al. 2003; Zuo et al. 2004; Jing et al. 2005; Karakasis and Stafylopatis 2008).

A GEP tagger is able to learn from a training corpus to produce an expression which observes the pattern of the

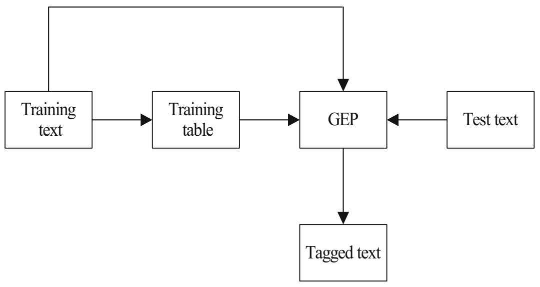

Fig. 2 Genetic expression programming train scheme

tags in corpus. Chromosome is generated from the tags in trained corpus. The evolution procedure remains the different contexts of each tag. The table can be computed by the training text and recording the different contexts and the number of occurrences of each of them for every tag in the training text.

Figure 2 shows the scheme of GEP training. The evolution process is run for each sentence in the text to be tagged. Evolution steps aim to maximize the total probability of the tagging of the sentences in the test corpus. The process finishes either if the fitness deviation lies below a threshold value or if the evolutionary process has been running for a maximum number of generations.

Evaluation of the performance of GEP has been undertaken, compared to various taggers. They compare the results on brown corpus of several different versions of intelligent algorithms: neural networks (Lippmann 1989; Schmid 1994; Marques and Lopes 2001) or genetic algorithms (Araujo 2002). They also give the performance of HMM method on Brown Corpus (Brants 2000).

The details of the brown corpus are as follows: the number of the words is 1,000,000; the number of sentences is 50,000; the number of tags is 80. The brown corpus was segmented into two parts, the training set of 90 % and the test set of 10 %, in the way that each sentence in the test set was extracted from every ten sentences.

From Table 2, it can be figured out that comparing with GA, neural network and HMM taggers the advantage of GEP can be obtained in a very efficient way in pos tagging. GEP tagger comes from GAs tagger, but is more efficient than it. The main disadvantage of GEP tagger is that it will spend a little time and spaces for data training just as GA and neural networks tagger do (Table 2). Accuracy rate on brown corpus.

Table 2 Comparison on brown corpus

Tagger type	Accuracy rate (%)
GEP	97.40
Neural networks	96.26
GA	96.0
HMM	96.62

3 Other taggers

The following are some other taggers worth be listing:

- Decision trees and statistical decision tree taggers produce output that is easier to interpret. A classical supervised algorithm of the machine learning field has been applied (Magerman 1995), in order to automatically acquire a language model for POS tagging based on statistical decision trees. This learning algorithm uses more complex contextual information than usual n-gram models and it can easily accept other kinds of information. However, critical for this type of tagger is the use of the correct set of questions in the decision part of the process (Màrquez et al. 2000).
- Finite state transducers seems a natural formalism to use for POS tagging which requires the sequential processing of inputs. In the context of POS tagging, the states represent the sequence of tags, and the output from the states is the words associated with the tags. See Sánchez-Villamil et al. (2004) for details.
- GAs tagger (Araujo 2002) use a genetic algorithm which, after the “evolution” of sequences of the taggers for the words in the text, select the best individual as solution. Gas tagger has developed a genetic algorithm that works with a population of potential tagging for each input sentence in the text. The evolution of individuals is based on a training table composed of contexts extracted from an in advance labelled training text.

4 Conclusions

POS tagging is a well-studied problem in natural language processing, in which the aim, given a natural language text, is to a label each word in that sample with a POS tag such as noun, verb or adjective. There are three main types of approaches to POS tagging: rule-based, stochastic methods and intelligent algorithm.

The feasibility of training computers to perform this task has been due to the development of annotated corpora, for example, the Brown corpus, Bank of English and the Penn Treebank. The annotations in a corpus provide ‘word–tag’ pairs which can be used to build a model and provide the expectations of which tag would accompany word w_i in the target corpus.

A typical rule-based method, TBL tagging is analysed. The TBL method uses the rules learned from the corpus and is a machine learning technique which takes a tagged corpus as input from which it can learn how to correctly tag a test sample. It reports an overall accuracy of 96.6 % on the Penn Treebank WSJ corpus using 900,000 words of training data.

Stochastic methods, more than rule-based methods, have used annotated corpora for POS tagging.

Two of the well understood and used stochastic methods were discussed: Markov models and SVM methods. These approaches and many others have performed with accuracies ranging from 96–97 %. It is believed that this is the level of accuracy that can be attained with the present annotated corpora due to annotation inconsistencies.

Rather new approaches with intelligent algorithm then are introduced. Net-Tagger and GEP tagger are detailed. Neural network and gene expression programming have shown their advantages on dressing the problem of machine learning and unknown things prediction. When they are applied on POS tagging, they can achieve a rather high accuracies (96.22 % for Net-Tagger, 97.4 % for GEP tagger).

As intelligent computing algorithms for POS tagging, a disadvantage of the intelligent methods may be their lower processing speed compared to statistical methods for their preceding training cost. In the light of the high speed of present computer hardware, however, this does not seem to be a serious drawback.

A number of opportunities exist for future research. It is possible that some of the modifications added which kept performance at an approximately constant level would actually result in better performance in downstream applications such as chunk parsing. Another area of potential research in this domain is that if unlexicalised tagging were considered, it is possible that new distinctions in the tag set could be far more productive, since the baseline tagger would have more valuable information.

Acknowledgments This work was supported in part by the National Natural Science Foundation of China (Nos. 61440018, 61501411), the Hubei Natural Science Foundation (No. 2014CFB904).

References

Araujo, L. (2001). Evolutionary parsing for a probabilistic context free grammar. In *Rough sets and current trends in computing, Canada* (pp. 590–597). Berlin: Springer.

Araujo, L. (2002). Part-of-speech tagging with evolutionary algorithms. In *Third International conference on computational linguistics and intelligent text processing*, *Mexico City*, *Mexico* (pp. 187–203).

Bohnet, B., & Nivre, J. (2012). A transition-based system for joint part-of-speech tagging and labeled non-projective dependency parsing. In *Joint conference on empirical methods in natural language processing & computational natural language learning*, *Jeju Island*, *Korea* (pp. 1455–1465).

Brants, T. (2000). TnT: a statistical part-of-speech tagger. In *Proceedings of the sixth applied natural language processing conference*, *Seattle*, *WA* (pp. 224–231). Trento: Association for Computational Linguistics.

Brill, E. (1992). A simple rule-based part of speech tagger. In *Proceedings of the third conference on applied computational linguistics* (pp. 112–116). Trento: Association for Computational Linguistics.

Brill, E. (1995). Transformation-based error-driven learning and natural language processing: a case study in part-of-speech tagging. *Computational Linguistics, 21*(4), 543–565.

Carlberger, J., & Kann, V. (1999). Implementing an efficient part-of-speech tagger. *Software-Practice and Experience, 29*(9), 815–832.

Charniak, E., Hendrickson, C., et al. (1993). Equations for part-of-speech tagging. In *AAAI-93, Proceedings* (pp. 784–784). New York: Wiley.

Cristianini, N., & Shawe-Taylor, J. (2000). *An introduction to support vector machines: and other kernel-based learning methods*. Cambridge: Cambridge University Press.

Cutting, D., Kupiec, J., et al. (1992). *A practical part-of-speech tagger* (pp. 133–140). Trendo: Association for Computational Linguistics.

Davis, M., & Dunning, T. (1995). Query translation using evolutionary programming for multi-lingual information retrieval. In *Proceedings of the fourth annual conference on evolutionary programming* (pp. 175–185).

Ferreira, C. (2001). Gene expression programming: a new adaptive algorithm for solving problems. Arxiv preprint cs/0102027.

Ferreira, C. (2003). Function finding and the creation of numerical constants in gene expression programming. In *Advances in soft computing*, 265.

Garrette, D., & Baldridge, J. (2013). Learning a part-of-speech tagger from two hours of annotation. In *Proceedings of NAACL*, Atlanta, Georgia (pp. 129–134).

Giménez, J., & Marquez, L. (2004). SVMTool: A general POS tagger generator based on support vector machines. In *Proceedings of the 4th international conference on language resources and evaluation* (*LREC'04*), Citeseer.

Goldberg, D. E. (1989). Genetic algorithms in search, optimization, and machine learning. Addison: Wesley.

Greene, B. B., & Rubin, G. M. (1971). Automatic grammatical tagging of English. Department of Linguistics, Brown University.

Jamatia, A., Gamblack, B., & Das, A. (2015). Part-of-speech tagging for code-mixed english-hindi twitter and facebook chat messages. In *Proceedings of recent advances in natural language processing* (pp. 239–248). Hissar.

Jing, P., Changjie, T., et al. (2005). M-GEP: a new evolution algorithm based on multi-layer chromosomes gene expression programming. *Chinese Journal of Computers, 28*(9), 1459–1466.

Karakasis, V. K., & Stafylopatis, A. (2008). Efficient evolution of accurate classification rules using a combination of gene expression programming and clonal selection. *IEEE Transactions on Evolutionary Computation, 12*(6), 662–678.

Karkaletsis, G., Petasis, G., & Paliouras, V. (2015). *Using machine learning techniques for part-of-speech tagging in the Greek language*. Singapore: World Scientific Publishing Company.

Kempe, A. (1993). A probabilistic tagger and an analysis of tagging errors. Rapport technique, Institut für maschinelle sprachverarbeitung, Universität stuttgart.

Krovetz, R. (1997). Homonymy and polysemy in information retrieval. In *Meeting of the Association for Computational Linguistics* (pp. 72–79). Trendo: Association for Computational Linguistics.

Lee, S. Z., Tsujii, J. I., & Rim, H. C. (2000). Lexicalized hidden markov models for part-of-speech tagging. In *International conference on computational linguistics* (pp. 481–487). Trendo: Association for Computational Linguistics.

Lippmann, R. P. (1989). Review of neural networks for speech recognition. *Neural Computation, 1*(1), 1–38.

Lv, C., Liu, H., et al. (2010). An efficient corpus based part-of-speech tagging with GEP. In *Sixth international conference on semantics, knowledge and grids* (pp. 289–292). IEEE.

Magerman, D. M. (1995). Statistical decision-tree models for parsing. In *Meeting of the Association for Computational Linguistics* (pp. 276–283). Trendo: Association for Computational Linguistics.

Manning, C. D., Schütze, H., et al. (1999). *Foundations of statistical natural language processing*. Cambridge: MIT Press.

Marques, N., & Lopes, G. (2001). Tagging with small training corpora. In *International symposium on advances in intelligent data analysis* (pp. 63–72). Berlin: Springer.

Màrquez, L., Padro, L., et al. (2000). A machine learning approach to POS tagging. *Machine Learning, 39*(1), 59–91.

Martinez, A. R. (2012). Part-of-speech tagging. *Wiley Interdisciplinary Reviews, 4*(1), 107–113.

Merialdo, B. (1994). Tagging English text with a probabilistic model. *Computational Linguistics, 20*(2), 155–171.

Nakagawa, T., Kudoh, T., et al. (2001). Unknown word guessing and part-of-speech tagging using support vector machines. In *Proceedings of the sixth natural language processing pacific rim symposium* (pp. 325–331).

Nakamura, M., Maruyama, K., et al. (1990). Neural network approach to word category prediction for English texts. In *International conference on computational linguistics* (pp. 213–218). Trendo: Association for Computational Linguistics.

Ngai, G., & Florian, R. (2001). Transformation-based learning in the fast lane. In *Proceedings of the second meeting of the North American Chapter of the Association for Computational Linguistics on Language Technologies* (pp. 1–8).

Owoputi, O., O'Connor, B., & Dyer, C. (2013). Improved part-of-speech tagging for online conversational text with word clusters. In *Proceedings of NAACL*, Atlanta (pp. 380–390).

Rabiner, L. R. (1989). A tutorial on hidden markov models and selected applications in speech recognition. In *Proceedings of the IEEE* (vol. 77(2), pp. 257–286).

Ratnaparkhi, A. (1996). A maximum entropy model for part-of-speech tagging. In *Proceedings of EMNLP'1996*, New Brunswick, New Jersey (vol. 1, pp. 133–142).

Sánchez-Villamil, E., Forcada, M., et al. (2004). Unsupervised training of a finite-state sliding-window part-of-speech tagger. *EsTAL, 2004*, 454–463.

Schmid, H. (1994). Part-of-speech tagging with neural networks. In *International conference on computational linguistics* (pp. 172–176). Trendo: Association for Computational Linguistics.

Smith, T. C., & Witten, I. H. (1995). A genetic algorithm for the induction of natural language grammars. In *Proc IJCAI-95 workshop on new approaches to learning for natural language processing* (pp. 17–24).

Sun, G., Lang, F., & Qiao P. (2008). Chinese part-of-speech tagging based on fusion model. In *Proceedings of the 11th joint conference on information sciences.* Amsterdam: Atlantis Press.

Thede, S. M., & Harper, M. P. (1999). A second-order hidden Markov model for part-of-speech tagging. In *Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics* (pp. 175–182).

Varile, G. B., & Zampolli, A. (1997). *Survey of the state of the art in human language technology*. Cambridge: Cambridge University Press.

Viterbi, A. (1967). Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory, 13*(2), 260–269.

Voutilainen, A. (2003). Part-of-speech tagging. The Oxford handbook of computational linguistics (pp. 219–232).

Wilks, Y., & Stevenson, M. (2000). Combining independent knowledge sources for word sense disambiguation. *Amsterdam Studies in the Theory and History of Linguistic Science Series, 4*, 117–130.

Tian, Y., & Lo, D. (2015). A comparative study on the effectiveness of part-of-speech tagging techniques on bug reports. In *International conference on software analysis, evolution and reengineering* (pp. 570–574). Montréal.

Zhou, C., Xiao, W., et al. (2003). Evolving accurate and compact classification rules with gene expression programming. *IEEE Transactions on Evolutionary Computation, 7*(6), 519–531.

Zuo, J., Tang, C., et al. (2002). Mining predicate association rule by gene expression programming. In *Advances in web-age information management* (pp. 281–294).

Zuo, J., Tang, C., et al. (2004). Time series prediction based on gene expression programming. In *Advances in web-age information management* (pp. 55–64).