Improving Part-of-speech Tagging for Ungrammatical Phrases

Daisuke Ninomiya

Supervised by Prof. Maxim Mozgovoy

Abstract

Modern part-of-speech (POS) tagging tools can provide high quality markup for grammatically correct documents, but ungrammatical sentences can be challenging for them. In this work we study the problem of POS tagging for the texts that contain grammatical errors, and show how POS-taggers can be improved for use in this context. Specifically, we propose to include ungrammatical POS-tagged sentences into the text corpus used to train a tagger (presumably, a tagger is based on a certain variation of machine learning).

Keywords

Part of speech tagging, natural language processing, ungrammatical phrases,

1 Introduction

Part of speech (POS) is one of the types into which words are divided in grammar according to their use, such as noun, verb, or adjective. POS tagging is to clarify a word and to annotate the POS of it. To be more specific, it is the process of marking up a word within a text as corresponding to a specific POS, based on its form, and context of larger syntactic frames like phrases, sentences, and paragraphs [1]. As for one word, if only one part of speech, there is not a problem. But English have words which have plural POS. For example,

1) I train tennis in the morning.
2) I like blue trains.

Figure 1: A word which have plural POS

"Train" of 1) is a verb. But "train" of 2) is noun. Although it is the same word, POS is different. Thus in order to decide on one POS, an algorithm becomes necessary. At first Show the example of the POS tagging in Figure 1.

I have a dog. ⟶	I	have	a	dog	.
	PRP	VBP	DT	NN	.

Figure 2: Example of POS tagging

POS tagging is an inevitable stage for a variety of natural language processing tasks. In particular, it is performed as a prerequisite for word sense disambiguation [2], parsing [3], and semantic classification [4]. Grammar checking software also can make use of part-of-speech markup. For instance, Language Tool — an extensible open source grammar checker allows the users to include POS tags into grammatical mal-rules 5].

The latter scenario, though, brings a new challenge. Most POS taggers are based on machine learning algorithms. In order to train and evaluate a tagger, one needs a POS-tagged text corpus, and typically such corpora consist of grammatically correct sentences only. However, in grammar checking we are also equally interested in the quality of POS tagging of sentences that contain grammatical errors.

This paper is dedicated to the problem of part of speech tagging in the domain of ungrammatical phrases. In the next section, explaining a background of POStagger. With section 3, analyzing the quality of existing part of speech taggers, then suggest and evaluate possible improvements through the inclusion of ungrammatical POS-tagged sentences into the training text corpus.

2 Background

As already mentioned, most today's part-of-speech taggers are typically based on machine learning. The tagger is first trained on a POS-tagged corpus, serving as a gold standard. Then the tagger uses obtained knowledge to markup unknown texts (Figure 3).

The taggers can implement different approaches to machine learning, such as support vector machines [6], maximum entropy modeling [7], decision trees [8], hidden Markov models [9], and so on.

A good training corpus should be unbiased, i.e. it should adequately represent possible input data. In practice, POS taggers are usually trained with well-known corpora, such as Brown [10] or OANC [10] for English. However, these corpora typically do not contain ungrammatical sentences, and thus they cannot be considered unbiased for the domain of computer-aided grammar checking.

Unfortunately, the same experiment of predecessors was not able to be found.

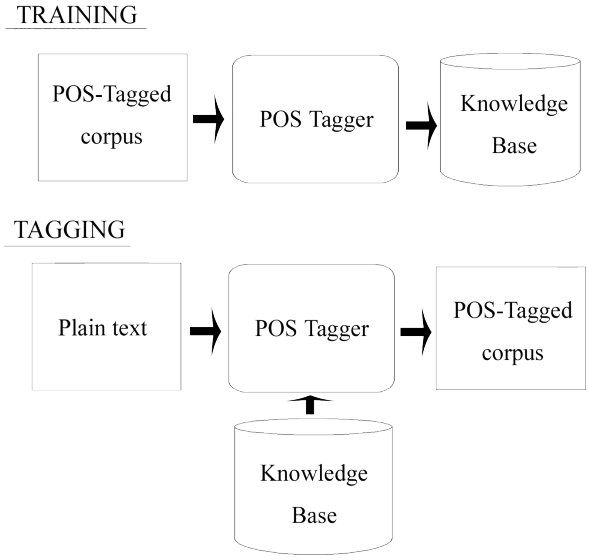

Figure 3: Flow of POS tagging

3 Experiments

3.1 Initial Step

For our experiments, we have selected two machine learning-based POS taggers: SVMTool and GCTag. SVMTool is an efficient POS tagger, based on support vector machines [6]. The second tagger, GCTag, is our own straightforward implementation of maximum entropy-based tagging scheme, similar to MXPOST [7] and based on open source Maximum Entropy Modeling (maxent) toolkit [12].

We used portions of manually POS-annotated Brown corpus [10] to train and test parsers. Our initial training set contains 260 179 tokens, and the test set consists of 412 075 tokens.

For this setup, the chosen POS taggers achieve the following levels of accuracy: SVMTool — 94.56%; GCTag — 91.11%. While SVMTool and GCTag exhibit lower accuracy than the current state-of-the-art projects, they do not implement ad-hoc and language-dependent tricks, thus providing reliable results for general evaluation of machine learning-based approaches to part-of-speech tagging.

3.2 Introducing Errors

Though text collections that contain ungrammatical sentences (error corpora) are well-known, they usually miss part-of-speech annotation. For the future experiments we plan to employ a range of artificial error creation methods [13, 14] to a number of POS-annotated texts in order to obtain a reasonable POS-annotated error corpus. However, within the present experiment we decided to concentrate on the following subset of selected simple errors.

Error Type 1: Wrong verb type. A verb of type VB (non-3rd-person singular present) is used instead of VBZ (3rd-person singular present) and vice versa.

Error Type 2: Wrong singular/plural form. A plural noun (NNS) is used instead of a singular noun (NN) and vice versa.

Error Type 3: Missing article. An article (a / the) is missing.

To create these errors, we wrote a computer program that performs the necessary substitutions (error types 1 and 2) and deletions (error type 3). For error types 1 and 2 we invoke AOT morphology analyzer and generator [14] in order to obtain desired verb and noun forms.

3.3 Tagging Accuracy of Modified Texts

First, let us evaluate the performance of part-of-speech taggers, trained with the initial (error-free) training set, on modified versions of the test set, containing errors of type 1-3. The results are summarized in Table 1.

Table 1: Tagging accuracy on modified test set

Error type	Tokens modified or deleted (%)	SVMTool accuracy (%)	GCTag accuracy (%)
no errors	0	94.56	91.11
1	3.59	92.70	89.49
2	17.26	94.09	90.04
3	8.19	93.62	89.63
1, 2	20.85	92.32	88.60
1, 2, 3	29.04	91.28	86.74

As we expected, biased training sets indeed cause degradation in the quality of POS tagging. Modified test sets contain contexts, never found in the training set. Consequently, the taggers become more prone to errors, regardless of their underlying algorithm (support vector machines or maximum entropy model).

3.4 Modifying Training and Test Sets

Now let us see how the inclusion of the same type 1-3 errors into the training set affects POS tagging. The experiments involve training the taggers on the following modified training sets:

1. (source set) ∪ (source set);
2. (source set) ∪ (source set with errors of type 1)
3. (source set) ∪ (source set with errors of type 2)
4. (source set) ∪ (source set with errors of type 3)
5. (source set) ∪ (source set with errors of types 1 and 2)
6. (source set) ∪ (source set with errors of types 1, 2, and 3)

We have doubled the source set for the first experiment in order to keep the same size of the training set for all sessions. We also decided to skip the experiments that involve modified source sets only, as we presume that in practice the tagger should be always able to process both grammatical and ungrammatical sentences (and thus it should be trained on the set that contains both erroneous and error-free texts).

Our new test sets are, however, not combined with the copies of the source test set, and include the following corpora: 1) source test set (without errors); 2) source test set with errors of type 1; 3) source test set with errors of type 2; 4) source test set with errors of type 3; 5) source test set with errors of types 1 and 2; 6) source test set with errors of types 1, 2, and 3.

3.5 Results

The accuracy values for our taggers, trained on all modified training sets, are shown in Table 2 and Table 3.

Table 2: SVMTool - Tagging accuracy evaluated with modified training and test sets

		Errors in the test set*					
		0	1 3.59	2 17.26	3 8.19	1, 2 20.58	1, 2, 3 29.04
Errors in the training set*	0	94.56	92.70	94.09	93.62	92.32	91.28
	1 3.66	92.45	94.69	91.74	91.34	93.93	93.0
	2 16.26	93.21	91.17	94.83	92.22	92.72	91.66
	3 9.10	85.38	83.82	85.29	93.80	83.47	91.42
	1, 2 19.92	91.38	93.35	92.76	90.28	94.79	93.93
	1, 2, 3 28.26	82.95	84.97	84.25	90.37	86.33	94.06

* Error type and the percentage of modified or deleted tokens

Table 3: GCTag - Tagging accuracy evaluated with modified training and test sets

		Errors in the test set*					
		0	1 3.59	2 17.26	3 8.19	1, 2 20.58	1, 2, 3 29.04
Errors in the training set*	0	91.11	89.49	90.04	89.63	88.60	86.74
	1 3.66	88.87	91.22	87.76	87.20	89.89	88.09
	2 16.26	89.81	88.13	91.58	88.25	89.75	87.79
	3 9.10	82.48	80.78	81.66	90.0	80.15	86.97
	1, 2 19.92	87.82	89.82	89.39	86.14	91.58	89.63
	1, 2, 3 28.26	79.39	81.38	80.84	86.52	83.02	89.99

* Error type and the percentage of modified or deleted tokens

3.6 Analysis

Unsurprisingly, the best results are most often achieved with training and test sets that match each other (i.e., when the training set and the test set contain the same types of errors). Likewise, the worst results are obtained in the experiments with most differences between training and test sets.

The accuracy of POS tagging for texts with errors is higher when the parser is trained with a training set that also contains ungrammatical sentences. This observation supports our initial idea: part-of-speech tagging for ungrammatical sentences can be improved through the inclusion of ungrammatical sentences into the training set. This technique works for both taggers evaluated in our experiments.

However, a modified tagger should be able to handle both grammatical and ungrammatical sentences. We have to admit that the inclusion of ungrammatical sentences into the training set causes significant reduction of accuracy when tagging error-free texts. Let us discuss the most valuable figures, obtained during the experiments (see Table 4).

Table 4: Accuracy of modified taggers for error-free and all-errors-included corpora

	Errors in the test set

			0	1, 2, 3
Errors in the training set	0	SVMTool GCTag	94.56 91.11	91.28 86.74
	1, 2, 3	SVMTool GCTag	82.95 79.39	94.06 89.99

As it can be seen, the taggers, trained with modified training sets, exhibit roughly the same performance on the test set with errors, as the original taggers on the original (error-free) test set. However, the modified taggers show noticeably lower accuracy on the error-free test set: the exclusion of errors in the test set causes considerable drop in accuracy from 94.06% to 82.95% in case of SVMTool, and from 89.99% to 79.39% in case of GCTag.

Interestingly, different error types in the training set cause significantly different degradation in tagging accuracy. Among our grammatical errors, error type 3 makes the largest contribution into this problem. For example, the inclusion of type 1 error into the training set decreases the accuracy of SVMTool on the error-free test set from 94.56% to 92.45%. Error type 2 causes less degradation (to 93.21%), and error type 3 provides considerable decrease of accuracy to 85.38%. Note that in our test set error type 3 is responsible for only 8.19% modified tokens, while for error type 2 this value is more than twice higher (17.26%). By keeping errors 1 and 2 only in the training set, we can obtain better figures (see Table 5).

Table 5: Accuracy of modified taggers for the selected training and test corpora

			Errors in the test set	
			0	1, 2, 3
Errors in the training set	0	SVMTool GCTag	94.56 91.11	91.28 86.74
	1, 2	SVMTool GCTag	91.38 87.82	93.93 89.63

This observation leads to the conclusion that for our experiments the training set containing error types 1 and 2 only can be considered as a reasonable trade-off.

4 Discussion and Conclusions

Part-of-speech tagging of ungrammatical sentences is an important sub-problem for a variety of natural language processing tasks, including computer-aided grammar checking. Most current part-of-speech taggers are not specifically designed to handle sentences that contain grammatical errors, and thus (as our experiments show) exhibit lower performance while processing ungrammatical phrases.

We have implemented and evaluated a straightforward way to tackle this problem, which consists in the use of error corpus (i.e. a POS-tagged text collection that contains ungrammatical sentences) for training a tagger.

Our experiments show that a tagger, trained with a combined corpus that contains both grammatical and ungrammatical sentences, is indeed able to tag ungrammatical text with high accuracy, comparable to the performance of the original tagger on error-free texts. However, this modification causes notable performance degradation in tagging error-free texts.

We have noted that this degradation highly depends on the types of errors, included into the training set. For instance, word form errors (incorrect verb and noun forms in our experiments) affect accuracy much less than missing words (omitted articles).

Therefore, we can conclude that our method is able to achieve the initial goal: to improve part-of-speech tagging for texts that contain grammatical errors. However, the inclusion of errors into the training corpora has to be done with care. Each type of grammatical fault should be tested separately in order to make sure that it does not cause significant loss of accuracy. In any case, the resulting tagger will exhibit trade-off performance: by improving accuracy of tagging erroneous sentences, we inevitably reduce the quality of processing error-free texts and vice versa.

5 Future Works

Because there were few error types and taggers which occurred this time, future research will likely increase the error type and the kinds of taggers. Thereby, a better result may be obtained by choosing the appropriate errors.

Acknowledgements

I thank for Professor Maxim Mozgovoy who was my supervisor. When there was not his help, I might not make it to here. I cannot thank you enough.

References

[1] Leech, G. (1997) Introducing corpus annotation. In, Garside, R., G. Leech and A. McEnery (Eds.) Corpus annotation: Linguistic information from computer text corpora. London: Longman, 1-18

[2] Moreno-Monteagudo, L., Izquierdo-Beviá, R., Martínez-Barco, P., and Suárez, A. 2006. A Study of the Influence of PoS Tagging on WSD. Lecture Notes in Computer Science, 4188, 173-179.

[3] Watson, R. 2006. Part-of-speech tagging models for parsing. Proceedings of the 9th Annual CLUK Colloquium, Open University, Milton Keynes, UK.
[4] Buitelaar, P., Ramaka, S. 2005. Unsupervised Ontology-based Semantic Tagging for Knowledge Markup. International Workshop on Learning in Web Search at ICML, 26-32.
[5] Naber, D. and Miłkowski, M. LanguageTool Development. Available at: http://www.languagetool.org/development (Accessed: 25.11.2011).
[6] Giménez, J. and Màrquez, L. 2004. SVMTool: A general POS tagger generator based on Support Vector Machines. Proceedings of the 4th International Conference on Language Resources and Evaluation, 43-46.
[7] Ratnaparkhi, A. 1996. A Maximum Entropy Part-of-Speech Tagger. Proceedings of the Empirical Methods in Natural Language Processing Conference, 133-142.
[8] Schmid, H. 1994. Probabilistic Part-of-Speech Tagging Using Decision Trees. Proceedings of International Conference on New Methods in Language Processing, vol. 12, 44-49.
[9] Brants, T. 2000. TnT — a Statistical Part-of-Speech Tagger. Proceedings of the 6th Conference on Applied Natural Language Processing (ANLP-2000), 224-231.
[10] Francis, W. 1980. A Tagged Corpus: Problems and Prospects. In: S. Greenbaum, G. Leech & J. Svartvik (eds.) Studies in English Linguistics for Randolph Quirk, 192-209.
[11] Macleod, C., Ide, N., and Grishman, R. 2000. The American National Corpus: Standardized Resources for American English. Proceedings of the 2nd Language Resources and Evaluation Conference (LREC), 831-36.
[12] Zhang Le. 2004. Maximum Entropy Modeling Toolkit for Python and C++. Natural Language Processing Lab, Northeastern University, China.
[13] Wagner, J., Foster, J., and van Genabith, J. 2007. A Comparative Evaluation of Deep and Shallow Approaches to the Automatic Detection of Common Grammatical Errors. Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, 112-121.
[14] Miłkowski, M. 2008. Automated Building of Error Corpora of Polish. Corpus Linguistics, Computer Tools, and Applications — State of the Art, 631-639.