Constraint-based Part-of-Speech Tagging

Neng-Fa Zhou
CUNY Brooklyn College & Graduate Center
zhou@sci.brooklyn.cuny.edu

This paper describes a constraint-based part-of-speech (POS) tagger, named CPOST, which treats POS tagging as a constraint satisfaction problem (CSP). CPOST treats each word as a variable, uses a lexicon to determine the domains of variables, employs context constraints to reduce ambiguity, and utilizes statistical models to label variables with values. This paper shows that, with a small number of context constraints that encode some of the basic linguistic knowledge, CPOST significantly enhances the precision at identifying base-form verbs, and mitigates the burden on syntax parsing.

1 Introduction

Part-of-speech (POS) tagging amounts to identifying the lexical type, which can be *noun*, *verb*, *adjective*, *adverb*, etc., of each word in a sentence. POS tagging is an important NLP task, and is a prerequisite to many downstream NLP tasks, such as syntax parsing, information extraction, and semantic parsing. Many words have multiple lexical types. For example, the word *can* serves as a modal verb in the sentence *We can do it*, but a noun in the sentence *I opened the juice can*. This type of ambiguity makes POS tagging a challenge.

The literature on POS tagging is abundant, and many POS taggers have been developed. Different approaches have been applied to POS tagging, including rule-based [3, 9, 7, 13], probabilistic models [2, 16, 17], neural network models [1, 14], and hybrid approaches [8, 11]. While the state-of-the-art POS taggers can achieve word accuracies of over 97%, sentence accuracies can hardly reach 60% [12]. On the sentence *what you listen to sounds amazing*, some of the popular POS taggers, including NLTK[1], spaCy[2], the Stanford POS tagger[3], and SyntaxNet[4] all misidentify the word *sounds* as a noun.

As the types of words often depend on the phrase structure of the sentence, it is not always possible to infer the types of words based on their local contexts. This paper proposes a constraint-based POS tagger, named CPOST, which treats POS tagging as a constraint satisfaction problem (CSP). CPOST treats each word as a variable, uses a lexicon to determine the domains of variables, employs context constraints to reduce ambiguity, and utilizes statistical models to label variables with values. A context constraint encodes some linguistic knowledge about uses of words. For example *a noun phrase cannot begin with a base-form verb* is a context constraint.

CPOST works hand-in-hand with a phrase structure parser, which is based on a relatively comprehensive grammar [4], utilizes backtracking and dynamic programming to efficiently search for parse trees, and adopts CHAT-80's rightmost attachment rule for resolving scoping ambiguity [15, 18]. For the above example sentence, *what you listen to sounds amazing*, the word *sounds* can be a noun or a verb. By working together with the parser, CPOST identifies the word *sounds* as a verb and helps the parser successfully parse the sentence.

[1] https://www.nltk.org/
[2] https://spacy.io/
[3] https://nlp.stanford.edu/software/tagger.shtml
[4] https://github.com/spoddutur/syntaxnet

Submitted to:
ModRef 2022

Admittedly, a comprehensive set of context constraints is required in order for CPOST to resolve all ambiguity, and even if such a set of constraints exists, it may take a lot of engineering effort to implement. This paper shows that, with a small number of context constraints that encode some of the basic linguistic knowledge, CPOST significantly reduces the ambiguity and mitigates the burden on the parser.

2 Preliminaries

CPOST uses a lexicon that is divided into the following subdictionaries: *verbs*, *nouns*, *adjectives*, *adverbs*, *pronouns*, *prepositions*, *determiners*, *modal verbs*, and *conjunctions*. The subdictionary of verbs maps a verb to its base form. For example, for the verb `see`, there is an entry in the subdictionary for *see* itself and each of the conjugated forms, including *sees*, *seeing*, *saw*, and *seen*. The subdictionary of nouns maps a noun to its base form. For example, for the noun `seed`, there is an entry for `seed` itself and an entry for the pluralized form `seeds`. The subdictionary of adjectives maps an adjective to its base form. For example, for the adjective `good`, there is an entry for *good* itself, an entry for its comparative, *better*, and its superlative, *best*. All other subdictionaries are sets of words. The lexicon is not required to be closed. All unknown words that do not occur in the lexicon are treated as nouns. Many words occur in multiple subdictionaries. For example, the word `can` occurs in three subdictionaries, including *modals*, *nouns*, and *verbs*. The task of POS tagging is to determine the lexical type of each word based on its context. This amounts to determining the subdictionary from which the word is taken.

The following set of lexical types is used in this paper:

CC	Conjunction (e.g., *if*, *and*, *or*)
DT	Determiner (e.g., *a*, *the*, *this*)
IN	Preposition (e.g., *for*, *on*)
JJ	Adjective
MD	Modal (e.g., *can*, *must*)
NJ	Noun or adjective
NN	Noun
PR	Pronoun (e.g., *we*, *its*)
PS	Possessive (e.g., *'s* in *John's book*)
RB	Adverb
SYM	Symbol
THERE	The word *there*
THAT	The word *that*
TO	The word *to*
VB	Verb

This set is an abstract of the set of Penn Treebank POS tags,[5] except that the type NJ is used as a super type of NN and JJ, and the word *that* itself is treated as a type.

This paper treats POS tagging as a constraint-satisfaction problem (CSP), where each word is treated as a variable, whose domain is the set of all possible lexical types allowed by the lexicon. Treating POS tagging as a CSP is reminiscent of treating type inference as a CSP in functional and logic programming languages.

Given a sentence, CPOST scans the words in the sentence from left to right, and generates context constraints. In this paper, $W^{(i)}$ indicates the ith word with respect to the current word, with $W^{(0)}$ indi-

[5]The name PS is used instead of the name POS originally used by Penn Treebank, as POS is used as the abbreviation for part-of-speech in this paper.

cating the current word, $W^{(-1)}$ indicating the previous word, $W^{(1)}$ indicating the next word, and so on. Similarly, and $T^{(i)}$ indicates the type of the ith word with respect to the current word.

The following predicates are used in the description of context constraints:

- `article(`W`)` is true if W is one of the articles: *a*, *an*, and *the*.
- `base(`W`)` is true if W is a base-form noun, verb, or adjective. For example, `base(can)`, `base(good)` and `base(see)` are true, but `base(sounds)` and `base(reduced)` are false.
- `be(`W`)` is true if W is a be-verb, meaning that W is one of the following: *am*, *are*, *be*, *been*, *being*, *is*, *was*, *were*.
- `bp(`W`)` is true if W is a base-form word or a plural noun: `bp(`W`)` $\leftrightarrow$ `base(`W`)` $\vee$ `plural(`W`)`.
- `phrasal_verb_of(`W`)` is true if W and the word *of* constitute a phrasal verb. For example, *conceive of* and *think of* are phrasal verbs.
- `plural(`W`)` is true if W is a plural form of a noun for which `base(`W`)` is false. For example, `plural(cans)` and `plural(leaves)` are true, but `plural(sheep)` is false because *sheep*'s plural form is identical to its base form.
- `ppn(`W`)` is true if W is a possessive pronoun. For example, `ppn(its)` and `ppn(her)` are true.
- `complementizer([`W_1`,`W_2`, ..., `W_n`])` is true if the word sequence W_1,W_2, …, W_n forms a complementizer [4]. For example, `complementizer([as,long,as])` and `complementizer([assuming, that])` are true.

3 Context Constraints

Context constraints specify linguistic knowledge about the types of words based on their local contexts. They are called *identifying characteristics* in [6]. Context constraints are utilized to constrain the assignment of types to words. This section gives constraints on nouns and verbs, with the focus on base-form words and plural nouns.

3.1 Context Constraints on Nouns

A noun phrase constitutes a context for a noun, which does not permit a verb in its base form. The following gives several constraints on BP words. Recall that `bp(`W`)` is true if W is a base-form word or a plural noun.

CN-1: T^{-1} `= JJ` $\wedge$ `bp(`W^0`)` $\rightarrow$ $T^0 \neq$ `VB`

If a BP word is preceded by an adjective (T^{-1} `= JJ`), then its type cannot be VB. For example, in *industrial conglomerate*, this constraint asserts that *conglomerate*'s type is not VB. Note that it is unsafe to force T^0 to be NN, as JJ can follow by JJ. For example, in *industrial average amount*, *average*'s type is JJ.

CN-2: T^{-1} `= PS` $\wedge$ `bp(`W^0`)` $\rightarrow$ $T^0 \neq$ `VB`

If a BP word is preceded by a possesion word (T^{-1} `= PS`), then its type cannot be VB. For example, in *company's jump*, *jump* cannot be VB if the suffix (*'s*) has the type PS.

CN-3: $\texttt{ppn}(W^{-1}) \wedge \texttt{bp}(W^0) \rightarrow T^0 \neq \texttt{VB}$

If a BP word is preceded by a possessive pronoun, as in *her company*, then its type cannot be VB.

CN-4: $\texttt{article}(W^{-1}) \wedge \texttt{bp}(W^0) \rightarrow T^0 \neq \texttt{VB}$

If a BP word is preceded by an article, as in *a company*, then its type cannot be VB.

CN-5: $\texttt{complementizer}([W^{-k}, \ldots, W^{-1}]) \wedge \texttt{bp}(W^0) \rightarrow T^0 \neq \texttt{VB}$

If a BP word is preceded by a complementizer, as in *when company*, then its type cannot be VB. CPOST looks back at *k* words in order to identify a complementizer, where *k* is 1, 2, or 3, depending on the length of the complementizer.

CN-6: $T^{-1} = \texttt{IN} \wedge \texttt{bp}(W^0) \rightarrow T^0 \neq \texttt{VB}$

If a BP word is preceded by a preposition ($T^{-1} = \texttt{IN}$), then its type cannot be VB. For example, in *at sports* and *on leave*, the words *sports* and *leave* cannot be assigned VB.

This constraint is taken from [6] (page 119). Note that the class IN is a superset of the Group-F words in [6]. Words, such as *although* and *than*, that are classified as IN by Penn Treebank, should be excluded as exceptions.

The word *to* can serve as a preposition when it is preceded by certain verbs (e.g., *according to*, *listen to*), nouns (e.g., *answer to*, *response to*), or adjectives (*kind to*, *cruel to*). [6] Note that this constraint cannot be applied if the sentence contains an extraposition, as in *what you listen to sounds amazing*, where *sounds* is clearly a verb despite that it follows the preposition *to*.

CN-7: $\texttt{be}(W^{-1}) \wedge \texttt{plural}(W^0) \rightarrow T^0 = \texttt{NN}$

If the current word is a plural form of a noun and is preceded by a be-word, as in *are concerns*, then its type is NN.

CN-8: $\texttt{bp}(W^0) \wedge W^1 = \texttt{of} \wedge \neg\ \texttt{phrasal\_verb\_of}(W^0) \rightarrow T^0 = \texttt{NN}$

If a BP word is followed by *of*, then the word cannot be a verb unless it is a phrasal verb together with `of`. For example, the word *yields* in *yields of* is not a verb.

Remarks

Note that in constraints CN-1 through CN6, the consequent is $T^0 \neq \texttt{VB}$. It's generally incorrect to change the consequent to $T^0 = \texttt{NN}$. For example, in *an even number*, it is incorrect to assign NN to *even*, as the word has four possible types, namely, VB, NN, JJ, and RB.

The constraints CN-1, CN-2, and CN-6 are entailment constraints that need to be generated and passed to the solver, while the other constraints can be enforced during preprocessing time. Consider again the example *an even number*. As the word *number* can be a noun or a verb, there are in total eight possible assignments of types to the words. While constraint CN-4 excludes VB from *even*'s domain during preprocessing time, constraint CN-1, which bans assigning (JJ, VB) to (*even*, *number*), must be maintained during solving time.

[6] https://dictionary.cambridge.org/us/grammar/british-grammar/to

3.2 Context Constraints on Verbs

A verb phrase, which is composed of at least one verb, is expected in certain contexts. This subsection gives several context constraints on base-form verbs, called infinitives.

CV-1: T^{-2} `= NN` $\wedge$ T^{-1} `= MD` $\rightarrow$ $T^0 \neq$ `NN`

If the current word is preceded by a modal verb (T^{-1} `= MD`), which is preceded by a noun (T^{-2} `= NN`), then its type cannot be NN. For example, in *stocks will jump*, *jump*'s type cannot be NN.

This constraint has several extensions. First, the modal verb can be followed by one or more adverbs, as in *stocks will eventually jump*. Second, a non-possessive pronoun can be treated as a noun, as in *it will jump*. Third, the verb *do* can also be treated as a modal verb, as in *stocks did not jump*.

This constraint is not valid if the modal verb expresses a condition, as in *I will buy should stocks rise*, where *stocks* clearly serves as a noun.

CV-2: `pre-infinitive(`W^{-2}`)` $\wedge$ W^{-1} `= to` $\rightarrow$ $T^0 \neq$ `NN`

If the current word is preceded by the word *to*, which is preceded by a *pre-infititive* verb, then its type cannot be NN. For example, in *he has to leave*, *leave* is not a noun. The *pre-infinitive* class includes many words, such as *want*, *love*, *hope*, and *attempt*. The modal verb *ought* also belongs to this class.

Remarks

Note that it would be unsafe for constraints CV-1 and CV-2 to force T^0 to be VB, as the word may be followed by an adverb, which happens to be a possible verb. For example, in *stocks will even jump*, it'd be wrong to assign VB to *even* as it serves as an adverb here.

4 Labeling Variables

After domain variables and constraints are generated, CPOST labels the variables from left to right with values based on the statistics obtained from a training corpus. Two statistical models are utilized to order values, namely a unigram model on ambiguous words and a trigram model on types. The unigram model gives P($T^0|W^0$), the probability of type T^0 given word W^0. The trigram model on types gives P($T^0|T^{-2},T^{-1}$), the probability of type T^0 given the previous two types T^{-2} and T^{-1}. The dummy type, `nil`, is used if a type is missing. So for a word that occurs first in a sentence, both T^{-2} and T^{-1} are `nil`, and for a word that occurs second in a sentence, T^{-2} is `nil` and T^{-1} is the type of the word preceding it. For the variable V^0 of the current word W^0 whose domain is D^0, CPOST first orders the values in D^0 based on the unigram model, from the most likely values to the least likely values, and then orders the values using the trigram model.

5 Implementation and Preliminary Experimental Results

CPOST requires a lexicon. Surprisingly, no such a lexicon is available that contains both base-form words and their inflections. There is a collection of 466k English words available.[7] However the words

[7] `https://github.com/dwyl/english-words`

Table 1: CPOST's performance on identifying BP verbs

Datasets	CPOST0		CPOST	
	Recall	Precision	Recall	Precision
TreeBank	89.5%	90.5%	92.2%	95.3%
CONLL2000	89.5%	86.7%	90.0%	92.6%

are not categorized. Using WordNet[8] and OPTED[9], which contain base-form words, and the rules for conjugating verbs, pluralizing nouns, and generating inflectional adjectives, this project categorized 134k of the words. Among the 134k words, 17k (14%) are ambiguous, meaning that they occur in more than one subdictionary.[10]

CPOST is implemented in the Picat language [20].[11] For a given sentence, it generates a domain variable for each word, whose domain is determined by the subdictionaries that contain the word. It scans the words in the sentence from left to right, and processes the context constraints. For constraints that can be fully processed, such as CN-3: $\texttt{ppn}(W^{-1}) \wedge \texttt{bp}(W^0) \rightarrow T^0 \neq \texttt{VB}$, it only excludes a nogood value from the domain of the current variable, and do not post the constraints into the constraint store. For constraints that cannot be fully processed, such as CN-1: $T^{-1} = \texttt{JJ} \wedge \texttt{bp}(W^0) \rightarrow T^0 \neq \texttt{VB}$, it adds the constraints into the constraint store.

Picat supports several constraint solvers. The CP solver is used in the implementation, as this is the only solver that allows user-defined procedures for ordering variables and values.

As CPOST only incorporates constraints on BP words, i.e., base-form words and plural nouns, this experiment evaluated how well it recognizes BP verbs in the TreeBank dataset[12], which contains 4010 sentences, and the CONLL 2000 dataset[13], which contains 10948 sentences. In this experiment, the unigram and trigram models used in labeling were obtained from the TreeBank dataset.

There are 4233 BP verbs in the TreeBank dataset. CPOST correctly recognizes 3904 of them, achieving a recall score of 92.2%, and misidentifies 190 BP words as verbs, achieving a precision score of 95.3% . There are 11480 BP verbs in the CONLL200 dataset. CPOST correctly recognizes 10330 of them, achieving a recall score of 90.0%, and misidentifies 829 BP words as verbs, achieving a precision score 92.6%. Table 1 shows recall and precision scores. For the sake of comparison, the table also gives the scores in column CPOST0 obtained by the version of the system that does not use any context constraints. It can be seen that both scores are significantly improved with the use of context constraints.

In terms of time efficiency, using context constraints only slightly slows down the system. On a Windows PC with an Intel i7 3.30GHz CPU and 64GB RAM, it took CPOST 8.46 seconds to tag the TreeBank dataset, amounting to 0.002 seconds per sentence, while it took CPOST0 7.59 seconds to tag the same dataset.

[8] `https://wordnet.princeton.edu/`

[9] `https://www.mso.anu.edu.au/~ralph/OPTED/`

[10] The dictionary is available at `https://github.com/nfzhou/english_dictionary`.

[11] `http://www.picat-lang.org/`

[12] `https://github.com/nlp-compromise/penn-treebank`

[13] `https://github.com/teropa/nlp/tree/master/resources/corpora/conll2000`

6 Discussion and Conclusion

Constraint-based POS tagging is related to both symbolic and statistical approaches to POS tagging. It treats POS tagging as a CSP, uses context constraints to constrain the assignment values to variables, and employs statistical models in labeling variables with values. This paper has presented a constraint-based POS tagger, named CPOST, and demonstrated that, when combined with statistical models, a small number of constraints can significantly improve the accuracy of POS tagging.

CPOST is different from Eric Brill's rule-based tagger [3, 13]. The rule-based POS tagger first assigns a likely tag to each word based on statistics, and then uses rules to refine the assignment. Constraints are different from Brill's rules in two aspects: (1) constraints are linguistically correct, while rules only make sense statistically; (2) constraints are non-directional, while rules are directional, meaning that the order in which rules appear or are applied have a big effect on the final assignment.

As long as POS tagging is concerned, constraints used in CPOST are similar to constraints used in constraint grammar (CG) [9, 10] in the sense that constraints are non-directional.[14] CG does not treat POS tagging as a CSP. It only uses constraints to *discard as many alternatives as possible*. In contrast, some of the constraints in CPOST are relations that need to be maintained during search for a viable parse tree.

Several tasks are on the stack for the future. Firstly, CPOST only incorporates several local context constraints on base-form verbs and plural nouns. Constraints need to be introduced for disambiguating the identification of other word types, such as participles of verbs, gerunds, and adjectives. Secondly, CPOST currently employs a simple statistical model in ordering domain values. It could be improved by adopting a sophisticated model, such as a neural network model, for ordering values. Thirdly, CPOST serves as a component of a logic-based phrase structure parser, and works hand-in-hand with the parser in disambiguating POS tagging. Certain global constraints are maintained by the parser. For example, a complete sentence must contain at least one verb, and no three base-form verbs can occur in a row. It needs to be evaluated how much the syntax parser can help improve the accuracy of POS tagging.

Acknowledgement

This work is supported in part by the NSF under the grant number CCF1618046.

References

[1] Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. Globally normalized transition-based neural networks. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*. The Association for Computer Linguistics, 2016.

[2] Michele Banko and Robert C. Moore. Part-of-speech tagging in context. In *COLING 2004, 20th International Conference on Computational Linguistics, Proceedings of the Conference, 23-27 August 2004, Geneva, Switzerland*, 2004.

[3] Eric Brill. A simple rule-based part of speech tagger. In *3rd Applied Natural Language Processing Conference, ANLP 1992, Trento, Italy, March 31 - April 3, 1992*, pages 152–155. ACL, 1992.

[4] Laurel J. Brinton and Donna M. Brinton. *The Linguistic Structure of Modern English*. John Benjamins Publishing Company, 2010.

[14] Constraints used in CG-3 [5, 19], a non-monotonic constraint grammar, are directional.

[5] Tino Didriksen. Grammar manual: 3rd version of the cg formalism variant. Grammarsoft aps, denmark, Yale Computer Science Report 1165, 2021.

[6] Charles Carpenter Fries. *The Structure of English*. Longmans, Green and Company, 1952.

[7] Donald Hindle. Acquiring disambiguation rules from text. In Julia Hirschberg, editor, *27th Annual Meeting of the Association for Computational Linguistics, 26-29 June 1989, University of British Columbia, Vancouver, BC, Canada, Proceedings*, pages 118–125. ACL, 1989.

[8] Mans Hulden and Jerid Francom. Boosting statistical tagger accuracy with simple rule-based grammars. In Nicoletta Calzolari, Khalid Choukri, Thierry Declerck, Mehmet Ugur Dogan, Bente Maegaard, Joseph Mariani, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Eighth International Conference on Language Resources and Evaluation, LREC 2012, Istanbul, Turkey, May 23-25, 2012*, pages 2114–2117. European Language Resources Association (ELRA), 2012.

[9] Fred Karlsson. Constraint grammar as A framework for parsing running text. In *13th International Conference on Computational Linguistics, COLING 1990, University of Helsinki, Finland, August 20-25, 1990*, pages 168–173, 1990.

[10] Fred Karlsson, Atro Voutilainen, Juha Heikkil, and Arto Anttila. *Constraint Grammar: A Language-Independent System for Parsing Unrestricted Text*. Mouton de Gruyter, 1995.

[11] Hongwei Li, Hongyan Mao, and Jingzi Wang. Part-of-speech tagging with rule-based data preprocessing and transformer. *Electronics*, 2021.

[12] Christopher D. Manning. Part-of-speech tagging from 97% to 100%: Is it time for some linguistics? In Alexander F. Gelbukh, editor, *Computational Linguistics and Intelligent Text Processing - 12th International Conference, CICLing 2011, Tokyo, Japan, February 20-26, 2011. Proceedings, Part I*, volume 6608 of *Lecture Notes in Computer Science*, pages 171–189. Springer, 2011.

[13] Grace Ngai and Radu Florian. Transformation based learning in the fast lane. In *Language Technologies 2001: The Second Meeting of the North American Chapter of the Association for Computational Linguistics, NAACL 2001, Pittsburgh, PA, USA, June 2-7, 2001*. The Association for Computational Linguistics, 2001.

[14] Dat Quoc Nguyen and Karin Verspoor. An improved neural network model for joint POS tagging and dependency parsing. In Daniel Zeman and Jan Hajic, editors, *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies, Brussels, Belgium, October 31 - November 1, 2018*, pages 81–91. Association for Computational Linguistics, 2018.

[15] Fernando C. N. Pereira. *Logic for Natural Language Analysis*. Research note 275, SRI International, 1983.

[16] Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. Feature-rich part-of-speech tagging with a cyclic dependency network. In Marti A. Hearst and Mari Ostendorf, editors, *Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics, HLT-NAACL 2003, Edmonton, Canada, May 27 - June 1, 2003*. The Association for Computational Linguistics, 2003.

[17] Sunita Warjri, Partha Pakray, Saralin A. Lyngdoh, and Arnab Kumar Maji. Part-of-speech (POS) tagging using conditional random field (CRF) model for khasi corpora. *Int. J. Speech Technol.*, 24(4):853–864, 2021.

[18] David H. D. Warren and Fernando C. N. Pereira. An efficient easily adaptable system for interpreting natural language queries. *Am. J. Comput. Linguistics*, 8(3-4):110–122, 1982.

[19] Anssi Yli-Jyrä. The power of constraint grammars revisited. *CoRR*, abs/1707.05115, 2017.

[20] Neng-Fa Zhou, Håkan Kjellerstrand, and Jonathan Fruhman. *Constraint Solving and Planning with Picat*. Springer Briefs in Intelligent Systems. Springer, 2015.