Incorporation of a Markov Model of Language Syntax in a Text Recognition Algorithm

Jonathan J. Hull
Center of Excellence for Document Analysis and Recognition
Department of Computer Science
State University of New York at Buffalo
Buffalo, New York 14260-0001
hull@cs.buffalo.edu

Abstract

A Markov model for language syntax and its use in a text recognition algorithm is proposed. Syntactic constraints are described by the transition probabilities between classes. The confusion between the feature string for a word and the syntactic classes is also described probabilistically. A modification of the Viterbi algorithm is also proposed that finds a fixed number of sequences of syntactic classes for a given sentence that have the highest probabilities of occurrence, given the feature strings for the words. An experimental application of this approach is demonstrated with a word hypothesization algorithm that produces a number of guesses about the identity of each word in a running text. It is shown that the Viterbi algorithm can significantly reduce the number of words that can possibly match an image.

1. Introduction

Text recognition algorithms often process only images of isolated characters. This is sometimes followed by a post-processing step that uses information from a dictionary of allowable words to correct recognition errors. Additional knowledge above the level of individual words can also be used to correct for recognition errors. An example includes the use of semantics of a constrained domain (chess games) to improve character recognition [1]. The transitions between pairs of words have also been used to improve a word recognition technique [7].

Language-level syntax has also been employed to improve word recognition by reducing the number of alternatives for the identity of a word [8]. This was done by using binary constraints between a group of words with the same shape and the syntactic classes that could follow them. The constraints were compiled from a training text and applied to restrict the decisions for the syntactic class of a word to be consistent with the shape of the previous word. Even this limited binary information was shown to be effective at reducing the average number of words that could match any image.

This paper proposes to model English grammar as a Markov process where the probability that any grammatical class will be observed is dependent on the class of the previous word.[1] This model is applied to text recognition by first using a word recognition algorithm to supply a number of alternatives for the identity of each word. The syntactic categories of the alternatives for the words in a sentence are then input to a modified Viterbi algorithm that determines sequences of syntactic classes that include each word. An alternative for a word decision is output only if its syntactic class included in at least one of these sequences. The Markov model improves word recognition performance if the number of alternatives for a word are reduced without removing the correct choice.

The rest of this paper briefly introduces an algorithm for word recognition. This is followed by a description of how a Markov model of language syntax is incorporated in this model. The modified Viterbi algorithm proposed in this paper is then described. The performance of this technique in reducing the number of alternatives for words in a sample of text is then discussed.

2. Text Recognition Algorithm

The recognition algorithm that incorporates the Markov model of syntax contains the three steps shown in Figure 1. The input is a sequence of word images w_i, $i = 1,2, \cdots$. The hypothesis generation stage computes a group of possible identifications for w_i (called N_i or its *neighborhood*) by matching a feature representation of the image to the entries in a dictionary. The hypothesis testing phase uses the contents of a neighborhood to determine a specific set of feature tests that could be executed to recognize w_i. The output of hypothesis testing is either a unique recognition of w_i or a set of hypotheses that contains the word in the image. The global contextual analysis phase uses information about other words that have been recognized, such as their syntactic classification, to constrain the words that can be in N_i. The output is a neighborhood N_i^* of reduced size.

3. Syntax Model

The syntax of a sentence is summarized as the sequence of syntactic classifications for its words. Since it is known that the appearance of any tag probabilistically constrains the tags that can follow it, a Markov model is a natural representation for syntax [10]. An example of such a probabilistic constraint is given by the probabilities that certain syntactic classes follow an article in a large sample of text. The word following an article is a singular or mass noun in 51 percent of all cases and is an adjective 20 percent of the time. The other 29 percent of occurrences are scattered over 82

[1] Probabilistic information has been successfully used to automatically tag large corpora of text [2].

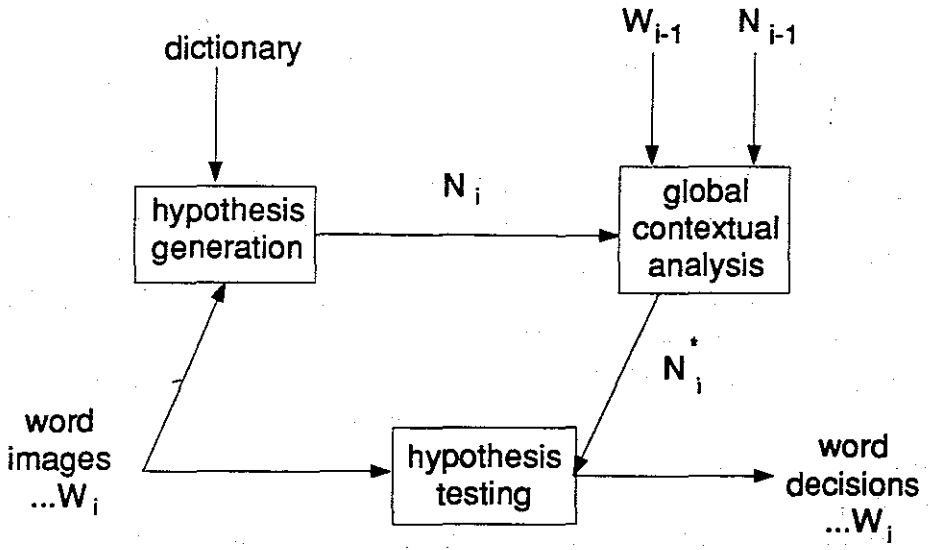

Figure 1. The text recognition algorithm.

other syntactic classes with many never following an article.

The Viterbi algorithm is a method for determining the sequence of classes with the maximum a-posteriori probability that match a given set of recognition decisions for the words in a sentence [4]. The adaptation of the Viterbi algorithm to this problem is very similar to its use in postprocessing character decisions [5]. The Viterbi algorithm has also been successfully used for speech recognition [3].

Under the assumptions that sentence delimiters (periods) are perfectly recognized and the occurrence of a word recognition error in any position is independent of the occurrence of an error in any other position, the Viterbi algorithm determines the string of syntactic classes $\bar{Z} = z_0, z_1, \ldots, z_{n+1}$ that maximize Bayes' formula:

$$P(\bar{Z} \mid \bar{X}) = \frac{P(\bar{X} \mid \bar{Z})\, P(\bar{Z})}{P(\bar{X})} \qquad 1$$

where $\bar{X} = x_0, x_1, \ldots, x_{n+1}$ is the sequence of feature vectors for the words in the sentence and $x_0 = x_{n+1} = z_0 = z_{n+1} = period$.

The independence and Markov assumptions, as well as the fact that the maximization of equation 1 is independent of $\bar{X}$ reduces the maximization of equation 1 to the maximization of:

$$\prod_{i=1}^{n+1} P(X_i \mid Z_i)\, P(Z_i \mid Z_{i-1}) \qquad 2$$

over all possible $\bar{Z}$, where $P(X_i \mid Z_i)$ is the conditional probability of the feature vector X_i taking on its value given that the corresponding syntactic class is Z_i, sometimes called a "confusion probability", and $P(Z_i \mid Z_{i-1})$ is the first order transition probability of the occurrence of syntactic class Z_i given that syntactic class Z_{i-1} has been observed. To avoid the combinatorial explosion inherent in calculating equation 2 for all possible strings $\bar{Z}$, the Viterbi algorithm uses a dynamic programming formulation to transform the problem into one of graph searching with a computational requirement on the order of M^2 operations, where M is the number of alternatives for any word image. The basic Viterbi algorithm is summarized:

Procedure Viterbi($\bar{X}$,$\bar{Z}$);

```
classes     := NULL;
prev_cost := 1;
for Word := 1 to Numwords+1 do { examine each word in sentence }
 begin
  for j := 1 to Num_alts_Word do { each alternative for current word }
   begin
    for k := 1 to Num_alts_Word-1 do { how can it be reached from prev. word }
S1:      temp[k] := prev_cost[k] * P(X_Word | Z_j) * P(Z_j | Z_k)
    partial_cost[j] := max(temp,u);
    temp_classes[j] := concat(classes[u], Z_j);
   end
  prev_cost := partial_cost;
  classes := temp_classes;
 end
write(classes[1]);
```

where, prev_cost and classes are vectors of real values and strings, respectively, that have the same number of elements as the maximum number of recognition decisions for any word. Prev_cost holds the accumulated costs of reaching the alternatives for the previous word and word holds the corresponding strings of syntactic tags. Max(temp,u) returns the maximum of the elements in temp with the side-effect that u is set to the index of the maximum element; concat(classes[u], Z_j) returns the result of concatenating Z_j on the end of classes[u]. The algorithm operates one word at a time from left to right in a sentence. The words are numbered 0,1,...,Numwords+1, where Numwords is the number of words in the input sentence. The string of syntactic classes on the best-cost path is output from classes[1] after the last iteration.

Symp. on Document Analysis and Information Retrieval, Las Vegas, NV, March 16-18, 1992

A useful modification of the Viterbi algorithm is to allow it to find a fixed number of syntactic tag sequences, each of which have the next best cost among all possible alternatives. This modification is simply implemented by maintaining the desired number of alternative sequences and their costs at each point in the evaluation. The final result includes those sequences and their costs.

4. Experimental Investigation

Experimental tests were conducted to determine the ability of the Viterbi implementation of syntactic constraints to reduce the number of word candidates that match any image. Given a sentence from a test sample of running text, a set of candidates for each word were produced by a model of the hypothesis generation portion of the word recognition algorithm. These candidates were looked up in a dictionary to retrieve their syntactic classes as well as their confusion probabilities. The Viterbi algorithm was then run on these data, using transition probabilities from another large text. Word candidates were then removed if their syntactic classes did not appear in any of the results produced by the Viterbi.

Performance was measured by determining the average number of words present in the neighborhoods before and after the application of syntax. The number of errors were also determined. An error occurred when the correct choice for a word was not present after the application of syntax.

An example of applying the Viterbi for syntactic constraints is shown in Figure 2. The original input sentence is shown along the the top of the figure (HE WAS AT WORK.). The complete neighborhoods for each word are shown below the input words. Each word is shown along with its syntactic class and the confusion probability for that neighborhood given the syntactic class. The different syntactic classes are explained in Appendix 1. The first and third words in the sentence have only one word in their neighborhoods. The second neighborhood contains eight different words, two of which have different syntactic classes. The fourth neighborhood contains six different words, one of which has three different syntactic tags.

The transition probabilities are shown along the arcs. It is seen that some transitions, such as PPS-NNS never occurred in the training text and hence have a probability of zero. Other transitions are much more likely, such as PPS-VBD which has a probability of 0.3621.

In this case, the top choice of the Viterbi algorithm was PPS-BEDZ-IN-NP and the second choice was PPS-BEDZ-IN-NN. In the top choice, three of the four classes encompassed the correct word. The correct answer for the fourth word (NN) was only contained in the second choice of the Viterbi.

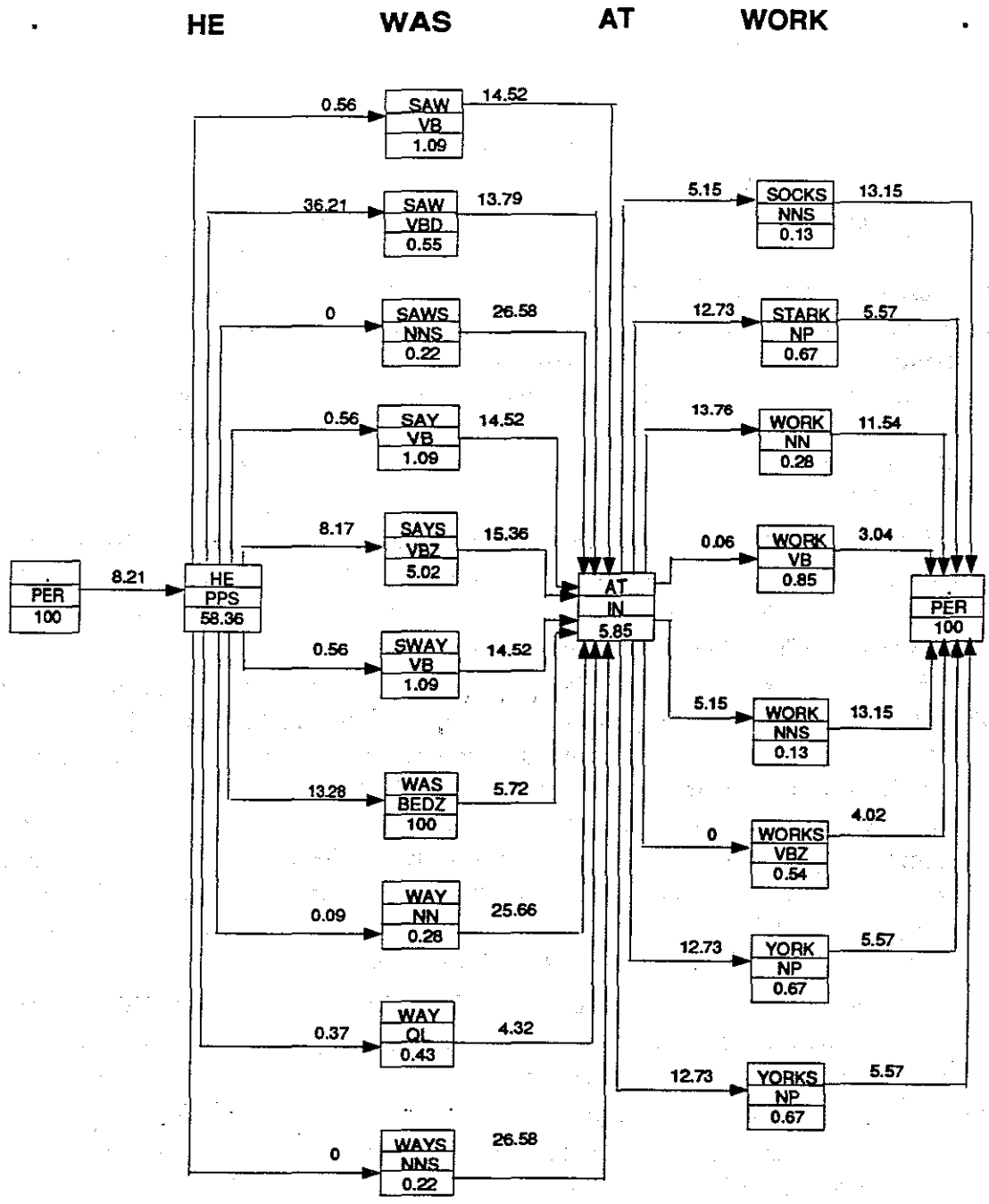

Figure 2. Example of applying the Viterbi algorithm .

Symp. on Document Analysis and Information Retrieval, Las Vegas, NV, March 16-18, 1992

4.1. Text Database

A soft copy (ASCII) text sample known as the Brown Corpus was used for these experiments [9]. This text was chosen because it is large (over 1,000,000 words of running text) and every word is tagged with its syntactic class. The corpus is divided into 15 subject categories or genres. There are 500 individual samples of running text in the corpus and each one contains approximately 2000 words. The number of samples in each genre differs depending on the amount published in that area at the time the corpus was compiled. The genres and the number of samples in each one are listed in Table 1.

The syntactic tags used to describe words in the corpus are organized in six major categories:

(1) *parts of speech:* nouns, verbs, adjectives, and so on.

(2) *function words:* determiners, prepositions, conjunctions, and so on.

(3) *important individual words: not*, existential *there*, infinitival *to*, forms of *do*, *be*, and *have*.

genre	category	no. samples
A.	Press Reportage	44
B	Press Editorial	27
C	Press Reviews	17
D	Religion	17
E	Skills	36
F	Popular Lore	48
G	Belles Lettres	75
H	Miscellaneous	29
J	Learned	80
K	General Fiction	29
L	Mystery and Detective Fiction	24
M	Science Fiction	6
N	Adventure and Western Fiction	29
P	Romance and Love Story	29
R	Humor	9

Table 1. The genres and the number of samples in each genre of the Brown Corpus.

(4) *punctuation marks of syntactic importance:* ".", "(", ")", "--", ",", ":".

(5) *inflectional morphemes:* noun plurals, and possessives, verb past, present and past participle, and so on.

(6) *foreign or cited words.*

A tag sometimes has a 'U' affixed to it to indicate the word is a negation. Altogether, 84 different tags were used in the experiments described in this paper. Those tags are listed in Appendix 1.

4.2. Hypothesis Generation Algorithm

The operation of the hypothesis generation algorithm was simulated by calculating the feature description for a word from pre-defined features for the letters in the word. All the words in a dictionary with the same feature description were used as the neighborhood for an input word.

The feature description is specialized for lower case characters because the experimentation is restricted to text written in lower case. The feature description includes vertical bars of different heights, dots, and empty spaces. These features were chosen because they can be reliably computed from images of text even if the characters touch one another [6]. The complete listing of this feature set is given below:

1. A significant area at the beginning or end of a word that does not contain a vertical bar (e.g., the space to the right of the vertical bar in a "c" or an "e");
2. A short vertical bar (e.g., the leg of an "r");
3. A long high vertical bar that extends above the main bar of the word (e.g., the ascender portion of a "b");
4. A long low vertical bar that extends below the main body of the word (e.g., the descender in a "p");
5. Dots over short vertical bars (occurs in an "i");
6. Dots over long vertical bars (occurs in a "j").

When the feature description is applied to a word it yields a symbolic representation that would correspond to the sequence of occurrence of the features in an image of the word. Thus, both "me" and "may" have a symbolic representation 22221 and the neighborhood of "me" is {"me", "may"}.

4.3. Experimental Application

The Viterbi algorithm for syntactic analysis was applied to genre A (newspaper reportage) of the Brown Corpus. The first sample in the genre (A01) was used as test data and the remainder as training data (A02-A44). The transition and confusion

probabilities were calculated from the training data. Because the words in the test data were assumed to be present in the image, the dictionaries were constructed from all of genre A.

Performance was measured by calculating the average neighborhood size per text word before and after the application of syntax. This statistic is defined as:

$$ANS_t = \frac{1}{N_w} \sum_{i=1}^{N_w} ns_i$$

where N_w is the number of words in the test sample and ns_i is the number of words in the neighborhood for the i^{th} word in the text. The error rate is the percentage of words with neighborhoods that do not contain the correct choice after the application of syntax.

In the example presented in Figure 2, N_w is 4, $ns_1 = 1$, $ns_2 = 8$, $ns_3 = 1$, and $ns_4 = 6$. $ANS_t = \frac{1}{4}$ (1+8+1+6) = 4.0. After the application of syntax, ANS_t was reduced to 1.75. This is an overall reduction of about 44 percent in ANS_t with a zero percent error rate.

The results of applying syntactic constraints with the Viterbi algorithm to the 88 sentences in A01 are shown in Table 2. The reduction in ANS_t and error rates that were incurred with between one and five alternatives from the Viterbi are shown. The average neighborhood size per text word is reduced by about 50% in all cases. Surprisingly, the error rate was very low, less than about 2 percent in every instance.

no. alternatives	ANS_t before	ANS_t after	% reduction	error rate
1	2.344	1.155	51%	2.20%
2	2.344	1.186	49%	1.48%
3	2.344	1.210	48%	1.17%
4	2.344	1.231	47%	0.87%
5	2.344	1.254	46%	0.76%

Table 2. Performance of syntactic constraints.

5. Discussion and Conclusions

An approach for incorporating syntactic constraints in a word recognition algorithm was presented. The recognition algorithm produced a series of choices for the words in a sentence. Syntactic constraints were used to remove some of these choices and thereby improve recognition performance.

Syntax was modelled as a first-order Markov source and the Viterbi algorithm was used to find the sequence of syntactic classes that best fit the choices provided by the recognition algorithm. A modification of the Viterbi algorithm provided several next-best alternatives for the syntactic class sequence as well as their costs.

An experimental investigation of this approach was performed in which a simulation of the word recognition algorithm was run on a test sample of text. It was shown that the average number of choices that could match a word can be significantly reduced by the proposed algorithm for applying syntactic constraints with a very low cost in error rate.

Future work on this approach will include the use of images to generate neighborhoods. More extensive experimentation with other portions of the corpus will also be performed. This will include consideration of second order transition probabilities as well as an interface to other syntactic information. The effect of loosening the restriction that the true word must appear in the lexicon will also be considered.

Symp. on Document Analysis and Information Retrieval, Las Vegas, NV, March 16-18, 1992

References

1. H. S. Baird and K. Thompson, "Reading Chess," *IEEE Transactions on Pattern Analysis and Machine Intelligence 12* (1990), 552-559.
2. A. D. Beale, "Lexicon and grammar in probabilistic tagging of written English," *Proceedings of the 26th Annual Meeting of the Association for Computational Linguistics*, Buffalo, New York, June 7-10, 1988, 211-216.
3. V. Cherkassky, M. Rao, H. Weschler, L. R. Bahl, F. Jelinek and R. L. Mercer, "A maximum likelihood approach to continuous speech recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI-5*, 2 (March, 1983), 179-190.
4. G. D. Forney, "The Viterbi algorithm," *Proceedings of the IEEE 61*, 3 (March, 1973), 268-278.
5. J. J. Hull, S. N. Srihari and R. Choudhari, "An integrated algorithm for text recognition: comparison with a cascaded algorithm," *IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI-5*, 4 (July, 1983), 384-395.
6. J. J. Hull, "Hypothesis generation in a computational model for visual word recognition," *IEEE Expert 1*, 3 (Fall, 1986), 63-70.
7. J. J. Hull, "Inter-word constraints in visual word recognition," *Proceedings of the Conference of the Canadian Society for Computational Studies of Intelligence*, Montreal, Canada, May 21-23, 1986, 134-138.
8. J. J. Hull, "Feature selection and language syntax in text recognition," in *From Pixels to Features*, J. C. Simon (editor), North Holland, 1989, 249-260.
9. H. Kucera and W. N. Francis, *Computational analysis of present-day American English*, Brown University Press, Providence, Rhode Island, 1967.
10. R. Kuhn, "Speech recognition and the frequency of recently used words: A modified Markov model for natural language," *Proceedings of the 12th International Conference on Computational Linguistics*, Budapest, Hungary, August 22-27, 1988, 348-350.

Appendix 1

Syntactic tags and Transition Probabilities

The syntactic tags used in the experiments.

(	left paren	BEZU	*isn't*	HVZU	*hasn't*	QL	qual
)	right paren	CC	coor. conj.	IN	preposition	QLP	post-qual
,	comma	CD	card. num.	JJ	adj	RB	adverb
--	dash	CS	sub. conj.	JJR	comp. adj	RBR	comp. adverb
:	colon	DO	*do*	JJS	sem. super. adj	RBT	super. adverb
ABL	pre-qualifer	DOD	*did*	JJT	morph. super. adj	RP	adverb/particle
ABN	pre-quant	DODU	*didn't*	MD	modal auxiliary	TO	infinitive marker *to*
ABX	pre-quant	DOU	*don't*	MDU	negative auxiliary	U	*not*,*n't*
AP	post-det	DOZ	*does*	NN	s. or mass n	UH	interjection, excl.
AT	article	DOZU	*doesn't*	NNS	pl. n	VB	verb, base form
BE	*be*	DT	det	NP	prop. n	VBD	verb, pst tense
BED	*were*	DTI	det/quant	NPS	pl. prop. n	VBG	verb, pres. part./ger.
BEDU	*weren't*	DTS	pl. det	NR	adverbial n	VBN	verb, pst part.
BEDZ	*was*	DTX	det/dbl conj.	OD	ordinal number	VBZ	verb, 3rd. pres.
BEDZU	*wasn't*	EX	exist. *there*	PN	noml. pro.	WDT	*wh* - det
BEG	*being*	HV	*have*	PP	pers. pro.	WP	*wh* - pro.
BEM	*am*	HVD	*had* pst tense	PPL	ref./int. pers. pro.	WPO	objective *wh* - pro.
BEN	*been*	HVDU	*hadn't*	PPLS	pl. ref./int. pers. pro.	WPS	nom. *wh* - pro.
BER	*are*,*art*	HVG	*having*	PPO	objective pers. pro.	WQL	*wh* - qual
BERU	*aren't*	HVN	*had* pst part.	PPS	3rd nom. pro.	WRB	*wh* - adverb

important abbreviations: adj = adjective, card. = cardinal, comp. = comparative, corr. = coordinating, det. = determiner, int. = intensive, n = noun, nom. = nominative, noml. = nominal, part. = participle, pers. = personal, pl. = plural, poss. = possessive, pro. = pronoun, prop. = proper, pst = past, qual. = qualifier, quant = quantifier, ref. = reflexive, s. = singular, sub. = subordinate, super. = superlative,