Hidden Markov Model for POS Tagging

Soren Larsen
UCSC Silicon Valley Extension
Santa Clara, California
snlarsen@ucsc.edu

1 Deriving the Viterbi Algorithm

1.1 Why Does $\arg\max_t P(t|w) = \arg\max_t \log P(t,w)$

To understand why maximizing $P(t|w)$ is equivalent to maximizing $\log P(t,w)$, we analyze the problem step-by-step:

1. Probability Maximization The goal is to find the tag sequence t that maximizes the conditional probability $P(t|w)$:

$$\arg\max_t P(t|w)$$

2. Log Transformation The logarithm is a monotonic function, meaning that for any two values a and b, if $a > b$, then $\log(a) > \log(b)$. Thus, maximizing $P(t|w)$ is equivalent to maximizing $\log P(t|w)$:

$$\arg\max_t P(t|w) = \arg\max_t \log P(t|w)$$

3. Bayes' Rule By Bayes' rule, the conditional probability $P(t|w)$ can be written as:

$$P(t|w) = \frac{P(w,t)}{P(w)}$$

where:

- $P(w,t)$ is the joint probability of the word sequence w and the tag sequence t.
- $P(w)$ is the marginal probability of the word sequence w, which is constant for all t.

Since $P(w)$ is constant, maximizing $P(t|w)$ is equivalent to maximizing $P(w,t)$:

$$\arg\max_t P(t|w) = \arg\max_t P(w,t)$$

4. Log-Space Representation To avoid numerical underflow when dealing with small probabilities, we perform computations in the log-space. Taking the logarithm:

$$\arg\max_t P(w,t) = \arg\max_t \log P(w,t)$$

This transformation simplifies computations, as probabilities that are multiplied in $P(w,t)$ become sums in $\log P(w,t)$:

$$\log P(w,t) = \log P(w|t) + \log P(t)$$

Thus, maximizing $P(w,t)$ or $P(t|w)$ is equivalent to maximizing $\log P(w,t)$, which is numerically stable and computationally efficient.

Conclusion Since the logarithm is monotonic and $P(t|w) \propto P(w,t)$, we conclude that:

$$\arg\max_t P(t|w) = \arg\max_t \log P(t,w)$$

1.2 Log Probability of the Highest Scoring Sequence

The log probability of the highest scoring tag sequence of length j ending in tag t_j, denoted as $\pi_j(t_j)$, is defined as:

$$\pi_j(t_j) = \max_{t_1,\ldots,t_{j-1}} \sum_{i=1}^{j} \text{score}(w,i,t_i,t_{i-1})$$

where the score function is:

$$\text{score}(w,i,t_i,t_{i-1}) = \log P(w_i|t_i) + \log P(t_i|t_{i-1})$$

To compute $\pi_j(t_j)$ recursively, consider the last transition in the sequence, from t_{j-1} to t_j. The best sequence up to position j ending in t_j must maximize:

$$\pi_j(t_j) = \max_{t_{j-1}} \left(\pi_{j-1}(t_{j-1}) + \text{score}(w,j,t_j,t_{j-1})\right)$$

Expanding the score function:

$$\pi_j(t_j) = \max_{t_{j-1}} \Big(\pi_{j-1}(t_{j-1}) + \log P(w_j|t_j) + \log P(t_j|t_{j-1})\Big)$$

Thus, $\pi_j(t_j)$ can be expressed as:

$$\pi_j(t_j) = \max_{t_{j-1}} \Big(\pi_{j-1}(t_{j-1}) + \log P(t_j|t_{j-1})\Big) + \log P(w_j|t_j)$$

This recursive formulation allows efficient computation of $\pi_j(t_j)$ for all t_j and forms the basis of the Viterbi algorithm.

1.3 Viterbi Algorithm and Complexity

This section provides the Viterbi Algorithm to compute $\hat{t}$, as defined in the last line of Eq. (1):

$$\hat{t} = \arg\max_t P(t|w)$$

The algorithm identifies the most probable sequence of tags for a given sequence of words. Additionally, the time complexity is analyzed to justify its efficiency.

1.3.1 Pseudocode for Viterbi Algorithm

Input:

- Observation sequence: $w[1:n]$ (words in the sentence)
- States: T (possible POS tags)
- Transition probabilities: $P(t_j \mid t_i)$ for all $t_i, t_j \in T$
- Emission probabilities: $P(w_k \mid t_i)$ for all w_k in Vocabulary and $t_i \in T$
- Initial probabilities: $P(t \mid \text{START})$ for all $t \in T$

Output:

- Most probable sequence of tags: $\hat{t}[1:n]$

Algorithm:

1. **Initialization:**
   - For each state $t \in T$:
   $$v[1][t] = \log P(t \mid \text{START}) + \log P(w[1] \mid t)$$
   - Set $\text{backpointer}[1][t] = \text{NULL}$
2. **Recursion:**
   - For $i = 2$ to n (iterate over each word):
   $$\begin{aligned} v[i][t] = \max_{t_{\text{prev}} \in T} & v[i-1][t_{\text{prev}}] \\ & + \log P(t \mid t_{\text{prev}}) \\ & + \log P(w[i] \mid t) \end{aligned}$$
   - Store the best previous state:
   $$\begin{aligned} \text{backpointer}[i][t] = \arg\max_{t_{\text{prev}} \in T} & v[i-1][t_{\text{prev}}] \\ & + \log P(t \mid t_{\text{prev}}) \end{aligned}$$
3. **Termination:**
   - Compute the final log-probabilities:
   $$\begin{aligned} v[n+1][\text{STOP}] = \max_{t \in T} & v[n][t] \\ & + \log P(\text{STOP} \mid t) \end{aligned}$$
   - Store the best previous state:
   $$\begin{aligned} \text{backpointer}[n+1][\text{STOP}] = \arg\max_{t \in T} & v[n][t] \\ & + \log P(\text{STOP} \mid t) \end{aligned}$$
4. **Backtrace:**
   - Initialize:
   $$\hat{t}[n+1] = \text{STOP}$$
   - For $i = n$ to 1:
   $$\hat{t}[i] = \text{backpointer}[i+1][\hat{t}[i+1]]$$
5. **Return:** $\hat{t}[1:n]$

1.3.2 Time Complexity

The time complexity of the Viterbi Algorithm is derived as follows:

- The recursion step computes $v[i][t]$ for every word $i = 1$ to n and every state $t \in T$. For each t, it iterates over all previous states $t_{\text{prev}} \in T$.
- Thus, the time complexity of the recursion is $O(n \cdot |T|^2)$, where n is the number of words and $|T|$ is the number of states.
- Initialization and termination contribute $O(|T|)$, and backtracing contributes $O(n)$. These are negligible compared to $O(n \cdot |T|^2)$.

Overall, the time complexity is:

$$O(n \cdot |T|^2)$$

This quadratic dependence on $|T|$ arises because every pair of current and previous states is considered. While computationally expensive for large $|T|$, this ensures globally optimal tag sequences.

1.3.3 Conclusion

The Viterbi Algorithm efficiently computes $\hat{t}$ by combining dynamic programming with backtracking. Its $O(n \cdot |T|^2)$ complexity is acceptable for small to moderate-sized tag sets and sequence lengths.

1.4 Semiring-Based Viterbi Algorithm

The Viterbi algorithm can be generalized using a semiring structure:

$$S = \langle A, \oplus, \otimes, 0_s, 1_s \rangle$$

where A is a set of elements, $\oplus$ and $\otimes$ are operations, and 0_s, 1_s are identity elements. In the Viterbi semiring:

- $\oplus = \max$: Selects the highest-scoring path.

- $\otimes = +$: Combines log probabilities.
- $0_s = -\infty$: Identity for $\oplus$.
- $1_s = 0$: Identity for $\otimes$.

The recursive computation of $\pi_j(t_j)$ becomes:

$$\pi_j(t_j) = \bigoplus_{t_{j-1}} (\pi_{j-1}(t_{j-1}) \otimes \text{score}(w, j, t_j, t_{j-1}))$$

where:

$$\text{score}(w, j, t_j, t_{j-1}) = \log P(w_j|t_j) + \log P(t_j|t_{j-1})$$

Semiring Properties The semiring-based formulation relies on:

- Associativity of $\oplus$ and $\otimes$: Ensures consistent combination of paths and scores.
- Distributivity of $\otimes$ over $\oplus$: Enables efficient recursion.
- Identity elements: $0_s = -\infty$ ensures missing paths are ignored, and $1_s = 0$ ensures scores add correctly.

Generalization Beyond Viterbi This semiring-based approach generalizes the Viterbi algorithm, allowing it to adapt to other tasks:

- For example, the Forward algorithm used for computing marginal probabilities can also be expressed using a semiring with $\oplus = +$ and $\otimes = \times$.
- Other semirings can encode variations, such as computing the k-best paths or summing over all paths instead of finding only the best one.

Equivalence with Standard Viterbi Algorithm When the max-sum semiring is used ($\oplus = \max$, $\otimes = +$), the semiring-based Viterbi algorithm is equivalent to the standard formulation:

$$\hat{t} = \arg\max_t \sum_{i=1}^{n} (\log P(w_i|t_i) + \log P(t_i|t_{i-1}))$$

This ensures compatibility with existing implementations while providing the flexibility to extend to other semirings.

Advantages of the Semiring-Based Approach The semiring generalization abstracts the algorithm's operations, offering:

- Flexibility to adapt to a wide range of problems, including Forward algorithms or variants for sequence labeling.
- Numerical stability when working with log probabilities, avoiding underflow during computation.
- A unified framework to reason about dynamic programming algorithms.

2 Implementation of the HMM-Based POS Tagger

2.1 Training the HMM Model

The training process involves estimating the transition and emission probabilities with add-α smoothing to address data sparsity. The following equations summarize the process:

- **Transition Probabilities:**

$$P(t_j \mid t_{j-1}) = \frac{\text{count}(t_{j-1}, t_j) + \alpha}{\text{count}(t_{j-1}) + \alpha \cdot |T|}$$

- **Emission Probabilities:**

$$P(w_i \mid t_j) = \frac{\text{count}(w_i, t_j) + \alpha}{\text{count}(t_j) + \alpha \cdot |\text{Vocab}|}$$

These probabilities are stored in log-space to avoid numerical underflow during calculations.

3 Viterbi Algorithm for Decoding

The Viterbi algorithm decodes the most probable sequence of tags for a given sequence of words. It uses dynamic programming with a backtracking mechanism to ensure optimal decoding. For a sequence of n words and m tags, the algorithm has a time complexity of $O(n \cdot m^2)$.

3.1 Baseline Model

A baseline model is implemented for comparison, assigning the most frequent tag for each word based on the training set. For unseen words, the default tag `NN` is used.

3.2 Debugging Example from Assignment PDF

A debugging routine validated the HMM and Viterbi algorithm using a toy example from the assignment. The example involved two words with predefined probabilities:

- Transition: $P(\text{VB} \mid \text{START}) = 4$, $P(\text{NN} \mid \text{VB}) = 9$, $P(\text{STOP} \mid \text{NN}) = 1$.
- Emission: $P(\text{example\_word1} \mid \text{VB}) = 1$, $P(\text{example\_word2} \mid \text{NN}) = 1$.

Results:

- True and Predicted Tags: `[VB, NN]`
- Gold and Predicted Scores: 3.58

The routine confirms the implementation accurately computes the optimal sequence and score.

This shorter version retains all essential details while reducing repetition and simplifying the explanation.

4 Experimental Results and Analysis

4.1 Hyperparameter Tuning

The smoothing parameter α was tuned using the development set. A grid search over $[0.1, 0.5, 1.0, 2.0]$ revealed that $\alpha = 0.1$ yields the best accuracy on the validation set (**88.97%**). This value was used for training the final model.

4.2 Performance Evaluation

The final HMM-based POS tagger was evaluated on the test set. Table 1 summarizes the results:

Metric	Value
Test Set Accuracy	89.16%
Baseline Accuracy	91.72%
Macro Precision	43.75%
Macro Recall	47.23%
Macro F1	43.46%

Table 1: Performance of HMM Tagger on the Test Set.

4.3 Confusion Matrix Analysis

The confusion matrix reveals that the HMM tagger performs well for high-frequency tags like `NN` (noun), `IN` (preposition), and `DT` (determiner). However, it struggles with systematic confusions:

- `NN` is often confused with `JJ` (848 cases) and vice versa (1147 cases).
- Plural nouns (`NNS`) are frequently mistaken for singular nouns (`NN`) in 470 instances.

These errors stem from overlapping syntactic contexts, e.g., "running" tagged as `NN` in "Running is fun" but as `VBG` in "He is running."

Performance declines further for low-frequency tags like `PDT` (predeterminer) and `FW` (foreign word), contributing to low macro-averaged metrics: precision (43.75%), recall (47.23%), and F1 (43.46%).

The full confusion matrix is provided in Appendix A.

4.4 Comparison with the Baseline Model

The HMM POS tagger achieves 89.16% accuracy, slightly below the baseline model's 91.72%. The baseline benefits from the dataset's skewed distribution, where frequent tags like `NN` dominate, enabling it to achieve high accuracy through simplicity.

While the HMM tagger generalizes better by incorporating transition and emission probabilities, it struggles with rare tags and exhibits systematic confusions.

Key points:

- HMM performs comparably to the baseline for frequent tags but struggles with rare ones.
- The baseline achieves higher accuracy but lacks flexibility for unseen data.

4.5 Improving the Tagger's Performance

To enhance the tagger's performance, several strategies can be employed:

- **Incorporating Contextual Features:** Using word embeddings like GloVe or contextual embeddings from BERT can provide richer representations of words, capturing syntactic and semantic nuances.
- **Enhanced Smoothing:** Implementing advanced smoothing techniques such as Good-Turing or Kneser-Ney smoothing can better address data sparsity for rare tags and transitions.
- **Semi-Supervised Learning:** Leveraging unlabeled data through pretraining or self-training methods can expand the model's coverage of linguistic contexts.
- **Ambiguity Resolution:** Annotating ambiguous sentences in the dataset with additional syntactic or semantic context can improve the model's ability to differentiate between tags like `NN` and `JJ`.

4.6 Optimizing Runtime Performance

The HMM POS tagger has a time complexity of $O(n \cdot |T|^2)$, where n is the sequence length and $|T|$ is the number of tags. The following optimizations can reduce runtime without significant accuracy loss:

- **Beam Search:** Limiting the number of tags considered at each step can drastically reduce computations while retaining high-probability paths.
- **Pruning Tag Space:** Reducing the number of tags by combining semantically similar tags into broader categories can decrease the computational burden.
- **Parallelization:** Distributing computations for transition and emission probabilities across multiple threads or GPUs can speed up the Viterbi algorithm.
- **Approximate Inference:** Using techniques like sampling-based inference or loopy belief propagation can reduce complexity with minimal accuracy trade-offs.

Appendix

A Detailed Confusion Matrix

The complete confusion matrix, detailing both the most frequently predicted tags and the most confused tag pairs, is attached as a text document in this submission. It provides valuable insights into the model's strengths and weaknesses for individual tags.

True Tag	Predicted Tag	Count
IN	NNP	156
NN	NN	21729
NNS	NNS	9477
VBG	VBG	1502
IN	JJ	354
,	,	9056
DT	DT	14450
IN	IN	16715
VBP	VBP	1523
VBN	VBN	2697
.	.	7035
CC	NNP	191
TO	TO	3898
VB	VB	4284
PDT	PDT	50
MD	MD	1651
IN	DT	150
VBP	IN	79
:	:	983
PRP	PRP	2769
JJ	JJ	8462
CC	CC	3847
EX	EX	168
VBZ	VBZ	3164
RB	RB	4330
RB	VB	22
JJR	JJR	395
``	``	1422
NN	JJ	848
JJ	NN	1147
VBG	NN	533
CD	CD	5337
NNS	NN	470
$	$	1138
DT	IN	34
NN	UH	28
''	''	1417
NNP	NNP	14711
VBG	JJ	365
WP	WP	392
VBD	VBD	4696
POS	POS	1638
NNS	NNP	167
VBD	VB	15
PRP$	PRP$	1402
RBR	RBR	169
RBR	JJR	104
WDT	WDT	713
WDT	DT	49

VBP	NN	201
WRB	WRB	422
VBD	VBN	773
NN	NNP	236
NNP	EX	79
RB	DT	26
CD	NNP	126
JJR	JJ	65
VBZ	POS	202
IN	NN	297
JJ	RB	205
NN	VBD	3
IN	WDT	122
PRP	DT	99
RB	NNP	74
DT	NNP	255
CC	IN	185
RB	JJ	426
NN	SYM	10
VB	NN	200
NNP	UH	115
NNS	DT	32
MD	NN	10
VBP	JJ	31
NNS	VBZ	259
VBP	VB	239
VBN	VBD	254
VBN	JJ	496
JJS	JJS	225
IN	RB	145
JJR	RB	6
WP$	WP$	47
VBD	JJ	178
VBN	VB	20
(	(	249
)	)	252
NN	VB	213
NN	VBG	31
MD	NNP	5
VB	VBP	111
VBZ	NNS	154
NNPS	NNP	172
VBZ	NNP	9
NNP	DT	191
NNP	NN	837
NNP	JJ	657
NNP	SYM	14
NN	PDT	11
VBD	DT	7
JJ	NNP	132

NNP	LS	35
VBZ	NN	16
VBZ	PDT	9
EX	RB	6
FW	FW	4
FW	NN	5
RBS	RBS	70
JJS	RBS	33
DT	JJ	5
PRP$	JJ	28
NN	.	17
RB	NN	255
NNS	LS	13
NNP	.	36
NNP	NNPS	20
IN	RBR	100
DT	RB	48
VBP	VBD	28
JJ	MD	79
CD	NN	186
VBG	PDT	28
NN	RBS	18
NNS	JJ	89
VBP	VBN	5
CC	DT	11
JJ	PRP	16
DT	WDT	60
NN	NN|JJ	3
RB	CC	13
JJ	IN	40
JJ	DT	278
WDT	WP	4
VBP	NNP	63
JJS	JJ	90
NN	DT	49
JJ	VBG	10
NNP	PDT	17
NNP	$	78
NNP	CD	91
VBG	NNP	12
RB	IN	510
RB	RP	47
NN	VBN	42
JJ	EX	9
VB	UH	5
JJ	PDT	30
NN	NNS	4
NNP	PRP	53
RB	MD	4
JJ	VB	29

VBG	IN	71
CD	NNS	17
NNP	MD	37
IN	RP	79
VB	RBS	14
NNP	RBS	70
JJ	CD	25
VBZ	RBS	7
RB	JJR	8
VBN	VBP	10
JJR	RBR	81
VBD	NN	127
JJ	.	5
NN	<STOP>	6
VBN	LS	6
CD	DT	77
DT	NN	60
JJ	$	22
NN	CD	18
CC	RB	13
NN	IN	17
NN	PRP	12
CD	RBS	29
VBZ	DT	8
VBD	VBP	19
RBR	NN	44
JJR	NNP	2
NNS	IN	10
JJR	VB	4
CD	UH	61
VB	VBN	19
WDT	IN	7
NNP	IN	29
NNS	VBN	35
VB	JJ	35
RBR	JJ	25
JJ	RBS	51
RB	RBS	20
NN	VBP	56
CD	EX	16
VBG	DT	17
VB	NNP	30
CD	PRP	9
CD	$	38
PDT	DT	13
RB	PDT	4
VB	$	3
NN	MD	33
NNP	VBG	2
JJ	UH	24

RB	SYM	1
PRP$	NNP	3
NN	EX	13
CD	,	3
FW	LS	2
NNP	NNS	43
IN	CC	19
PDT	CD	1
JJ	LS	18
VBG	RBS	16
RB	WP	4
NNPS	NNPS	8
VBD	RB	2
VBN	RB	12
NNS	NN|NNS	1
CD	JJ	12
UH	IN	1
NNPS	NNS	31
CD	LS	41
VBG	VBN	16
NNP	<STOP>	22
NNS	UH	21
NNPS	NN	6
RB	LS	7
RBR	JJS	2
NN	LS	34
NNS	CD	18
PRP	PRP$	8
VBG	<STOP>	1
RB	$	3
VBN	CD	1
JJ	SYM	5
PRP	VBP	14
CC	JJ	25
NN	,	1
VBN	NN	28
NN	$	9
PRP	NN	13
RB	PRP	3
VBP	UH	27
WRB	PDT	1
RB	VBP	11
RB	RBR	22
NN	RB	21
VBD	PDT	18
VBZ	VBN	19
VBN	NNP	11
JJR	NN	6
CC	NN	6
JJS	NNP	13

VBG	PRP	2
CD	WP$	2
PRP	NNP	14
VBN	DT	14
PRP	CD	4
NNS	RBS	23
NNPS	.	3
NNS	VB	5
FW	.	3
FW	<STOP>	4
#	#	22
PDT	JJ	1
JJ	VBN	77
FW	JJ	2
VBG	$	7
NNP	VBD	2
NNS	.	15
VBN	<STOP>	1
CD	.	6
FW	IN	2
VBP	RBS	4
RB	EX	13
VBD	WRB	1
NNP	VB	39
DT	PDT	28
RBS	NNP	4
VBD	RBS	6
NNP	WDT	5
VBD	IN	7
VBN	RBS	6
CD	VB	37
VBG	VBG|NN	3
CD	SYM	2
VBN	PDT	7
CC	SYM	9
NNP	:	9
WP	WDT	5
VB	.	2
NNS	MD	19
DT	CD	2
FW	NNP	9
JJ	VBP	8
NNPS	PRP	2
NNP	VBN	18
JJ	FW	1
JJS	RB	4
VBD	NNP	9
VB	PDT	26
JJS	NN	3
NNPS	MD	1

NNP	PRP$	1
RBS	JJ	8
JJ	WP$	5
NN	TO	1
RB	UH	42
PRP	FW	6
IN	FW	4
VBG	WP$	3
VB	``	1
VBP	PDT	2
NNPS	VBN	2
JJR	DT	2
NNS	SYM	9
NNP	JJS	3
RP	IN	236
PRP	POS	3
LS	LS	12
VB	DT	3
VBN	MD	5
VBN	UH	3
UH	UH	15
VB	LS	2
SYM	SYM	11
JJR	VBP	1
NNS	PRP	8
VBN	,	1
NNP	RBR|JJR	1
JJR	JJS	11
NNS	RB	8
RB	VBG	1
DT	,	1
MD	VBP	2
VBP	RB	2
VBZ	.	2
JJ	<STOP>	2
VBZ	VBD	2
VBZ	WRB	1
NN	WP	1
NNP	RB	3
JJ	NNS	6
UH	JJ	1
VBZ	UH	3
NNS	NNPS	5
NNS	<STOP>	4
VBN	SYM	1
JJS	MD	4
RBR	RB	31
NNPS	DT	2
NNPS	VBP	1
VB	IN	13

VBZ	IN	5
JJR	MD	3
CD	VBN	8
NNPS	RBS	1
CC	FW	1
NNPS	UH	3
NNPS	<STOP>	1
NNPS	CD	1
NNPS	LS	2
JJR	SYM	1
JJS	EX	1
RP	JJ	37
JJ	VBD	12
VB	VBD	9
RP	RB	19
RBS	JJS	4
NNP	VBP	2
RP	VBP	1
JJR	RBR|JJR	3
VBN|JJ	JJ	1
RP	NN	25
RP	RP	35
NNS	EX	3
NNP	VBZ	3
WRB	RB	1
RP	RBR	2
MD	VB	3
FW	PRP	1
FW	CD	2
FW	UH	2
VBG|NN	NN	1
FW	RB	1
VBG	UH	1
NN	WDT	1
IN	VB	1
VBN	PRP	2
DT	LS	2
MD	VBD	3
VBG	MD	5
VBD	.	3
PDT	NN	1
''	POS	6
NNP	JJR	1
NNPS	$	1
VBD	<STOP>	2
RB	VBZ	1
RBR|JJR	JJR	1
RB	VBN	3
JJR|RBR	RBR	2
NN	WP$	1

JJR|RBR	JJ	1
VBD	WP$	1
JJR	LS	1
VB	,	1
NNPS	JJ	2
FW	$	1
IN	VBP	1
VBG	.	3
CC	CD	1
VBG	LS	4
VBZ	NNS|VBZ	1
VBZ	MD	1
JJ|IN	PDT	1
JJ	RB|JJ	1
VBN	$	2
RB	CD	2
JJ	RP	2
CD	IN	10
IN	IN|RB	2
VBN|JJ	VBN	1
VB	WP$	1
VB	MD	1
VB	(	1
RB|JJ	NN	1
RB|JJ	UH	1
VBP	.	1
JJR|RBR	JJR	1
RBR|JJR	RBR	1
DT	SYM	1
VBP	EX	1
UH	DT	1
VBZ	$	4
VBD	CD	2
NNS	WP	2
TO	JJ	1
JJ	RBR	1
VBP	``	1
LS	NNP	3
UH	LS	1
NNP	``	1
RBR	VB	4
CD	MD	4
VBG	SYM	1
VB	JJR	2
IN	$	1
VBD	(	1
NNS	CC	1
NNS	PDT	1
VBN	.	2
WRB	DT	1

JJS	$	1
UH	NN	1
NNS	$	2
VBP	CD	1
VBD	UH	1
VBP	DT	1
CD	PDT	2
VBD	$	1
VB	PRP	2
JJ	JJR	2
IN	PDT	1
NNP	RP	1
NNP	POS	10
RP	IN|RB	1
NN	JJS	1
VB	RB	1
VBN	WP$	1
RBS	RB	1
RBR	VBP	2
VBG	EX	2
NN	VBG|NN	1
RBR	RBR|JJR	1
RB	RBR|JJR	1
VBZ	LS	1