HMM-HPS: A Heuristic Preseed Approach to Part-of-Speech Tagging with Hidden Markov Models

Michael Yantosca
Department of Computer Science, University of Houston / Houston, TX
mike@archivarius.net

Abstract

In spite of their simplicity, Hidden Markov Models can provide surprisingly accurate predictions for complex tasks in computational linguistics such as part-of-speech tagging. By applying heuristic functions against the given observations, augmenting the observed feature vector with the results, and accordingly preseeding the probability matrices of the model, accuracy can be substantially improved even when training against small corpora.

1 Introduction

Part-of-speech (POS) tagging serves a crucial role in the natural language processing pipeline as it provides a syntactic skeleton without which proper semantic analysis downstream would prove extremely difficult. However, the process of tagging is labor intensive, and as corpora have grown, the desire for expediting the process has driven development of automated POS taggers. Early implementations employed rule-based methods, but such approaches tended not to scale in the general case.

Alternative approaches using Hidden Markov Models (HMMs) to model a generative probability distribution whereby state transitions between the different parts of speech emit lexical observations have proven reasonably accurate. To explore the exact nature of the efficacy of HMMs in POS tagging and what further augmentations might prove beneficial, a HMM model was built from scratch, trained, tested, and refined in the *COSC 6336 Part-of-Speech Tagging with HMM* competition on `competitions.codalab.org` under the user handle `ghostant`.

2 Methodology

Training Initial development of the POS tagger began with building a frequency counter for arbitrary order n-grams encountered over the course of reading the training file. These n-grams included the n pairs of words and corresponding language identification tags up to the current word being read. As each new word was encountered, each order of n-gram leading up to the word was counted. Preceding words and language ID tags were kept in a bounded cursor to minimize processing overhead. If the cursor was full when it encountered a new word, the oldest entry was excised so that the new entry could be introduced and tallied accordingly.

Additionally, the count of each annotated POS tag was recorded along with the count of the transition from the previously seen POS tag. At the beginning of a sentence, an implicit `Q0` tag served as the origin of the transition, and at the end of a sentence, an implicit `QF` tag served as the terminus of the transition. The explicit tag definitions in the training corpus employed the Universal Dependencies tagset (Nivre et al., 2018).

Once the training file had been read and the frequency model established, the frequency counts were converted to probabilities. At first, Laplace smoothing (Jurafsky and Martin, 2017, 47) with equiprobable weights was used to supply probability mass for out-of-vocabulary words and unseen state transitions, but this was ultimately generalized to add-k smoothing (Jurafsky and Martin, 2017, 49) to permit tuning based on empirical results. Other smoothing methods were briefly considered but rejected on account of the time involved to implement them given the schedule of the competition.

No attempt was made to store the model in a form that could be loaded later, so training was done online in the same execution run as the testing. Whereas this significantly slowed development velocity at the beginning, the addition of later refinements and the stream-processing approaches employed in both training and testing made this an acceptable cost.

It should be noted that all training was done with the provided `train.conll`. On account of time constraints, more sophisticated methods of cross-validation (Jurafsky and Martin, 2017, 86-87) were not employed.

Testing The testing phase employed the Viterbi decoding algorithm as given in Jurafsky and Martin (2017, 131-134). To save on computation time and memory usage, a running a_{max} value was kept in the innermost loop for determining each state's most likely predecessor.

To gauge efficacy of the tagger during development and enable rapid prototyping, a rough measure of precision was taken via the following shell command:

```
diff dev.conll <output file> | egrep '<' | wc -l
```

All development testing prior to submission was done with the provided `dev.conll`, and this drove decisions on refining and tuning the model. Submissions to the competition were only trained with `train.conll`.

Refinements The first pass of the tagger pipeline revealed that it would not be able to compete effectively on its own. Rough estimates of precision from initial assays against `dev.conll` ran around 0.81, below the Naive Bayes competition baseline. It was noticed that many clearly nominal or verbal tokens were being labeled as `PUNCT`, so a heuristic fallback step was introduced in the decoding phase for unknown vocabulary. It employed regular expressions to capture whether the unigram in the emission was capitalized, contained only punctuation marks, or only alphanumeric characters. For capitalized unigrams, the fallback emission probability was zeroed out for every class except `PROPN`. For punctuation-only unigrams, the fallback emission probability was zeroed out for every class except `PUNCT`. For alphanumeric-only unigrams, the fallback emission probabilities for $P(word|\texttt{PUNCT})$ and $P(word|\texttt{SYM})$ were zeroed out.

The refinement performed as expected, raising the correctness of the output, if not perfectly preserving the probability masses involved. Taking a cue from Brants *et al.* in their development of *stupid backoff* (2007, 859-860), this was deemed an acceptable sacrifice. Following their terminology, the quasi-probabilities of the model are referred to hereafter as scores. Unfortunately, adding this refinement also incurred a massive performance penalty since the series of branching if-statements became quadratic with respect to Q, the set of possible generating POS states.

Attention then turned to combining the emission scores of different orders of n-grams. Two modes of operation were implemented to achieve this: backoff and interpolation (Jurafsky and Martin, 2017, 49-50). The backoff mode simply deferred to increasingly lower orders of n-grams from the highest order stored by the model until reaching the unigram order. If the model had not seen the token, it would fall back to a special token encoded as `<UNK>` that had been seeded with smoothed scores if one of the heuristic fallback cases did not apply.

The interpolation mode was implemented in a rudimentary fashion. The highest order n-gram would receive a weight of 0.5, the (n-1)-gram would receive half of the remaining weight, and so on until the unigram observation. If any of the n-grams had not been seen, its contribution would defer to the heuristic fallback unigram of `<UNK>`. At the beginnings of sentences and in sentences shorter than the highest order n-gram in the model, no phantom n-grams were computed. The interpolation chain would start with the longest observed n-gram and apply the formula in the same manner.

An issue was encountered in the dev set with a sentence composed of 91 tokens. A numerical underflow in score multiplication led to a state where the backpointers could not be traversed since they would only be assigned if a preceding state had a score greater than zero, the initial value of the a_{max} cursor. To counter this, a major refactoring of the code was done to store and use the logarithm of the various scores calculated, and multiplication operations were changed to addition. To provide the same effect in log space, the value zero was replaced with $-$`sys.float_info.max`.

Since the poor execution speed of the heuristic fallback put a drag on development time, the heuristic intuitions were integrated earlier in the pipeline by preseeding the emission scores that corresponded to the various heuristic checks. Three heuristic check responses were encoded in each observation key, and the regular expression match that provided the key was calculated once upon ingestion during the decoding phase. The three features that were added to the word and language ID as part of the observation were as follows:

- `SOME_WORD`: at least one alphanumeric character
- `CAP`: capitalized alphanumeric followed by alphanumeric and underscore characters
- `ALL_PUNCT`: only punctuation characters

This provided a significant performance boost that enabled development to progress more swiftly while maintaining the same level of precision. Some attempts were made to redistribute the probability mass appropriately, but in most cases heuristic downgrades and upgrades during the conversion of frequencies to scores were done unilaterally for the sake of time.

Furthermore, the iterative decay of significance in the weights for the rough interpolation of multiple orders of n-grams was modified to give more credence to higher-order n-grams. Instead of $\frac{1}{2}$ of the remaining weight, each n-gram in the chain received $\frac{3}{4}$ of the remaining weight.

As a final step, emission scores for POS states `INTJ`, `UNK`, and `X` were downgraded for unknown words that matched the `SOME_WORD` feature. This was done as a result of observations that these tags seemed to be disproportionately assigned in pre-submission test runs on `dev.conll`. The decision was made in the full understanding of the risk of overfitting on the presumption that unknown "dictionary" words would be far more likely to turn up in a corpus than novel interjections owing to the former's richer semantic content. This intuition is borne out when taking a cursory glance at other corpora (Davies, 2018) used in POS tagging research (Garside et al., 2007) (Brysbaert et al., 2012). For example, sampling every twentieth word from the list of 100,000 words based on the 450 million word Corpus of Contemporary American English (COCA) yields a mere four interjections in the set of 5,000, the most frequent being "goodbye" with a frequency ranking of 9,200 out of 100,000 (Davies, 2012).

3 Experimental Results

3.1 Competition Submissions

Results from submissions to the competition from training on `train.conll` and testing on `test.conll` were as follows[1]:

Accuracy	n	k	Heuristic
0.9181	1	1	fallback
0.9352	1	0.01	fallback
0.9347	2	0.05	preseed
0.9349	1	0.01	preseed
0.9355	4	0.01	preseed, revised weights

3.2 Local Experimentation

While the competition submissions exhibited decent overall accuracy, a more thorough examination of the model's performance in terms of accuracy, precision, recall, and F_1-measure (Jurafsky and Martin, 2017, 83-84) controlling for certain parameters would provide a more incisive inquiry into the strengths and weaknesses of this hybrid heuristic HMM approach. In the interest of ensuring clarity in the derivation of these measures, the values of word count (WC), gold positive and negative (GP and GN), true positive and negative (TP and TN), and false positive and negative (FP and FN) are defined on a per-class basis:

```
WC = $(egrep -v '^$' dev.conll | wc -l)
GP = $(egrep 'class$' dev.conll | wc -l)
GN = WC − GP
FN = $(diff dev.conll ../hw1-results/dev4.txt |
       egrep '<' | egrep 'class$' | wc -l)
FP = $(diff dev.conll ../hw1-results/dev4.txt |
       egrep '>' | egrep 'class$' | wc -l)
TP = GP − FP
TN = GN − FN
```

The per-class statistics were composited using both micro-averaging and macro-averaging. The composite results are given in the following graphs and tables.

Precision The micro-averaged precision matches very nearly the rough performance characteristic that drove development. Precision decreased as the constant used for add-k smoothing increased, which is to be expected given that the scores for heretofore unseen observations were equiprobable except in cases where they were overridden by heuristic measures. The performance of the system lags well behind

[1]The first submission used regular scores. The rest used log-space.

even simple naive Bayes systems when heuristic methods are not employed, but using the weighted interpolation with a small k constant seems to minimize the occurrence of false positives.

Micro-averaged Precision

n	k	bfb	bps	bnh	wfb	wps	wnh
1	$1 \cdot 10^{-3}$	0.93	0.94	0.8	0.93	0.94	0.81
1	$1 \cdot 10^{-2}$	0.93	0.94	0.79	0.93	0.94	0.8
1	0.1	0.93	0.93	0.81	0.93	0.93	0.79
1	1	0.92	0.92	0.77	0.92	0.92	0.77
2	$1 \cdot 10^{-3}$	0.9	0.9	0.74	0.93	0.94	0.81
2	$1 \cdot 10^{-2}$	0.9	0.9	0.77	0.93	0.94	0.79
2	0.1	0.9	0.9	0.76	0.93	0.93	0.79
2	1	0.86	0.86	0.7	0.92	0.92	0.81
3	$1 \cdot 10^{-3}$	0.88	0.89	0.74	0.93	0.94	0.81
3	$1 \cdot 10^{-2}$	0.88	0.89	0.77	0.93	0.94	0.83
3	0.1	0.88	0.88	0.72	0.93	0.93	0.81
3	1	0.83	0.83	0.63	0.91	0.91	0.74
4	$1 \cdot 10^{-3}$	0.88	0.88	0.75	0.93	0.94	0.81
4	$1 \cdot 10^{-2}$	0.88	0.88	0.76	0.93	0.94	0.79
4	0.1	0.88	0.88	0.72	0.93	0.93	0.78
4	1	0.83	0.83	0.61	0.91	0.91	0.77

Conversely, the macro-averaged precision values are significantly lower. The holistic precision of the system may be well-suited to the task and particularly the training and dev sets, but these results suggest that a few predominant classes in which the system performs well are masking the poor performance in the case of less frequently occurring POS tags.

Macro-averaged Precision

n	k	bfb	bps	bnh	wfb	wps	wnh
1	$1 \cdot 10^{-3}$	0.87	0.87	0.8	0.87	0.87	0.81
1	$1 \cdot 10^{-2}$	0.87	0.87	0.79	0.87	0.87	0.8
1	0.1	0.87	0.87	0.8	0.87	0.87	0.79
1	1	0.84	0.84	0.75	0.84	0.84	0.74
2	$1 \cdot 10^{-3}$	0.83	0.84	0.74	0.87	0.88	0.81
2	$1 \cdot 10^{-2}$	0.83	0.84	0.75	0.87	0.87	0.79
2	0.1	0.83	0.83	0.74	0.87	0.87	0.79
2	1	0.76	0.76	0.64	0.84	0.84	0.76
3	$1 \cdot 10^{-3}$	0.82	0.82	0.74	0.87	0.88	0.8
3	$1 \cdot 10^{-2}$	0.82	0.82	0.75	0.87	0.87	0.81
3	0.1	0.81	0.81	0.72	0.87	0.87	0.79
3	1	0.74	0.74	0.6	0.83	0.83	0.71
4	$1 \cdot 10^{-3}$	0.81	0.82	0.74	0.87	0.88	0.8
4	$1 \cdot 10^{-2}$	0.82	0.82	0.74	0.87	0.87	0.8
4	0.1	0.81	0.81	0.72	0.86	0.87	0.78
4	1	0.73	0.73	0.58	0.83	0.82	0.72

Recall The results for micro-averaged recall approximate the results observed for micro-averaged precision. Taken as a whole, the system does not seem to be excessively prone to false negatives.

Micro-averaged Recall

n	k	bfb	bps	bnh	wfb	wps	wnh
1	$1 \cdot 10^{-3}$	0.93	0.94	0.8	0.93	0.94	0.8
1	$1 \cdot 10^{-2}$	0.94	0.94	0.79	0.94	0.94	0.8
1	0.1	0.93	0.93	0.81	0.93	0.93	0.79
1	1	0.92	0.93	0.77	0.92	0.93	0.77
2	$1 \cdot 10^{-3}$	0.9	0.9	0.74	0.93	0.94	0.81
2	$1 \cdot 10^{-2}$	0.9	0.9	0.76	0.93	0.94	0.78
2	0.1	0.89	0.9	0.76	0.93	0.93	0.79
2	1	0.86	0.86	0.68	0.92	0.92	0.82
3	$1 \cdot 10^{-3}$	0.88	0.88	0.74	0.93	0.94	0.77
3	$1 \cdot 10^{-2}$	0.88	0.88	0.75	0.93	0.94	0.83
3	0.1	0.88	0.88	0.71	0.93	0.93	0.81
3	1	0.84	0.84	0.61	0.92	0.92	0.74
4	$1 \cdot 10^{-3}$	0.88	0.88	0.73	0.93	0.94	0.8
4	$1 \cdot 10^{-2}$	0.88	0.88	0.76	0.93	0.94	0.78
4	0.1	0.88	0.88	0.72	0.93	0.93	0.78
4	1	0.83	0.83	0.61	0.91	0.91	0.76

Once again, however, the macro-averaged recall indicates that there are certain underrepresented classes where the occurrence of false negatives is high

Macro-averaged Recall

n	k	bfb	bps	bnh	wfb	wps	wnh
1	$1 \cdot 10^{-3}$	0.86	0.87	0.82	0.86	0.87	0.8
1	$1 \cdot 10^{-2}$	0.87	0.87	0.8	0.87	0.87	0.8
1	0.1	0.87	0.87	0.81	0.87	0.87	0.8
1	1	0.87	0.87	0.79	0.87	0.87	0.79
2	$1 \cdot 10^{-3}$	0.82	0.84	0.78	0.86	0.87	0.81
2	$1 \cdot 10^{-2}$	0.83	0.84	0.77	0.86	0.87	0.81
2	0.1	0.83	0.83	0.76	0.87	0.87	0.8
2	1	0.8	0.8	0.68	0.87	0.87	0.8
3	$1 \cdot 10^{-3}$	0.81	0.82	0.76	0.86	0.87	0.79
3	$1 \cdot 10^{-2}$	0.81	0.82	0.76	0.86	0.87	0.81
3	0.1	0.82	0.82	0.72	0.87	0.87	0.79
3	1	0.77	0.77	0.62	0.86	0.87	0.76
4	$1 \cdot 10^{-3}$	0.8	0.82	0.76	0.86	0.87	0.8
4	$1 \cdot 10^{-2}$	0.81	0.82	0.75	0.86	0.87	0.8
4	0.1	0.81	0.81	0.73	0.87	0.87	0.79
4	1	0.77	0.77	0.62	0.86	0.86	0.76

Accuracy No distinction is made between micro-averaging and macro-averaging of accuracy since the results are mathematically the same.

Accuracy

n	k	bfb	bps	bnh	wfb	wps	wnh
1	$1 \cdot 10^{-3}$	0.99	0.99	0.98	0.99	0.99	0.98
1	$1 \cdot 10^{-2}$	0.99	0.99	0.98	0.99	0.99	0.98
1	0.1	0.99	0.99	0.98	0.99	0.99	0.98
1	1	0.99	0.99	0.97	0.99	0.99	0.97
2	$1 \cdot 10^{-3}$	0.99	0.99	0.97	0.99	0.99	0.98
2	$1 \cdot 10^{-2}$	0.99	0.99	0.97	0.99	0.99	0.98
2	0.1	0.99	0.99	0.97	0.99	0.99	0.98
2	1	0.98	0.98	0.96	0.99	0.99	0.98
3	$1 \cdot 10^{-3}$	0.99	0.99	0.97	0.99	0.99	0.98
3	$1 \cdot 10^{-2}$	0.99	0.99	0.97	0.99	0.99	0.98
3	0.1	0.99	0.99	0.97	0.99	0.99	0.98
3	1	0.98	0.98	0.96	0.99	0.99	0.97
4	$1 \cdot 10^{-3}$	0.99	0.99	0.97	0.99	0.99	0.98
4	$1 \cdot 10^{-2}$	0.99	0.99	0.97	0.99	0.99	0.98
4	0.1	0.99	0.99	0.97	0.99	0.99	0.97
4	1	0.98	0.98	0.95	0.99	0.99	0.97

The results observed here appear superficially impressive, but the abnormally high values point to an overfitting in the model. This becomes immediately apparent when one compares the individual values of the accuracy in the different classes. For instance, the `PUNCT` class boasts of accuracy measures as high as 0.9997 with heuristic provisions, but only 0.8782 without fallback or preseeding. And yet, the overall accuracy even without heuristic support is well above the 97th percentile in all cases. The empirical results here corroborate Jurafsky and Martin (2017, 83-84) in decrying accuracy as a useful statistic.

F_1-Measure Since F-measure is a function of precision and recall, the results are similar for those observed for precision and recall but are included here for completeness. It succinctly captures in one measure the inherent weaknesses of the model.

Micro-averaged F_1-measure

n	k	bfb	bps	bnh	wfb	wps	wnh
1	$1 \cdot 10^{-3}$	0.93	0.94	0.8	0.93	0.94	0.8
1	$1 \cdot 10^{-2}$	0.93	0.94	0.79	0.93	0.94	0.8
1	0.1	0.93	0.93	0.81	0.93	0.93	0.79
1	1	0.92	0.92	0.77	0.92	0.92	0.77
2	$1 \cdot 10^{-3}$	0.9	0.9	0.74	0.93	0.94	0.81
2	$1 \cdot 10^{-2}$	0.9	0.9	0.77	0.93	0.94	0.78
2	0.1	0.9	0.9	0.76	0.93	0.93	0.79
2	1	0.86	0.86	0.69	0.92	0.92	0.81
3	$1 \cdot 10^{-3}$	0.88	0.89	0.74	0.93	0.94	0.79
3	$1 \cdot 10^{-2}$	0.88	0.89	0.76	0.93	0.94	0.83
3	0.1	0.88	0.88	0.72	0.93	0.93	0.81
3	1	0.84	0.84	0.62	0.91	0.91	0.74
4	$1 \cdot 10^{-3}$	0.88	0.88	0.74	0.93	0.94	0.8
4	$1 \cdot 10^{-2}$	0.88	0.88	0.76	0.93	0.94	0.79
4	0.1	0.88	0.88	0.72	0.93	0.93	0.78
4	1	0.83	0.83	0.61	0.91	0.91	0.76

Macro-averaged F_1-measure

n	k	bfb	bps	bnh	wfb	wps	wnh
1	$1 \cdot 10^{-3}$	0.86	0.87	0.79	0.86	0.87	0.78
1	$1 \cdot 10^{-2}$	0.87	0.87	0.77	0.87	0.87	0.77
1	0.1	0.87	0.87	0.78	0.87	0.87	0.77
1	1	0.85	0.86	0.73	0.85	0.86	0.73
2	$1 \cdot 10^{-3}$	0.83	0.84	0.73	0.86	0.87	0.79
2	$1 \cdot 10^{-2}$	0.83	0.83	0.74	0.87	0.87	0.77
2	0.1	0.83	0.83	0.73	0.87	0.87	0.76
2	1	0.78	0.78	0.62	0.85	0.85	0.76
3	$1 \cdot 10^{-3}$	0.81	0.82	0.73	0.86	0.87	0.77
3	$1 \cdot 10^{-2}$	0.81	0.82	0.73	0.87	0.87	0.79
3	0.1	0.81	0.81	0.69	0.87	0.87	0.77
3	1	0.75	0.75	0.57	0.84	0.85	0.7
4	$1 \cdot 10^{-3}$	0.81	0.82	0.72	0.86	0.87	0.78
4	$1 \cdot 10^{-2}$	0.81	0.82	0.72	0.87	0.87	0.77
4	0.1	0.81	0.81	0.69	0.87	0.87	0.76
4	1	0.75	0.75	0.56	0.84	0.84	0.71

4 Conclusions

While the system performed relatively favorably in the competition, the faults in the system strongly recommend a refactoring of certain components. The small training set size commended itself to the use of heuristics, but better performance on a purely probabilistic basis might have been achieved with more comprehensive training data. A more rigorous examination of the proper redistribution of probability mass is required at a minimum.

Until the final refinements, the system was most precise with an observation order of 1, i.e., with unigrams. The original expectation was that increasing the observation order would increase the precision of the system, but the sparsity of the higher-order n-grams may have rendered increasing the order parameter of the model useless for all but the most common idioms.

Decreasing the k smoothing constant increased precision during development, but the strategy may not apply in cases with a large unknown vocabulary. The equiprobable distribution engendered by naive add-k smoothing fails to reflect real-life distribution of POS tags, demanding the addition of heuristic support. A more sophisticated approach with a cross-validated battery of tests against a collection of held-out corpora likely would have generated better empirical weights for not only unknown vocabulary but also improved interpolation beyond the simple iterative remainder method that was employed.

Furthermore, future efforts would likely do well to attempt a morphological deconstruction of observations during training since lexemes are not necessarily atomic. Key components of function are encoded in all manners of inflection which could be leveraged to provide more granular feature determination.

For all of the issues with the implementation as presented, there is an even more fundamental weakness in the HMM approach in general. The Markov assumption that the probability of a whole sequence can be estimated by immediate predecessors fails to account for situations in which subsequent context determines the syntactic relationship of words heretofore, such as in languages that employ postpositions, or for situations in which vital antecedent context precedes the current state by more than what may be economical to store, such as in topic-prominent languages.

To this end, it may be worthwhile to investigate strategies that employ lookahead tactics like skip-grams or to transfer language ID encoding into the state transition matrix rather than the observation matrix to better account for cases where the syntax of the two or more languages in a code-switched corpus follow a different transitional graph, such as the adjective-noun noun-adjective dichotomy between English and Spanish. Enriching the POS transition graph may even yield some insights, at least on a syntactic level, into the nature of code-switching itself.

Acknowledgments

The author wishes to acknowledge Dr. Thamar Solorio for heuristic feature suggestions in lecture on February 28, 2018; the NAACL template authors for providing a ready-made LaTeX template; and Dr. Daniel Jurafsky and Dr. James H. Martin for succinctly expositing Hidden Markov Models in their text *Speech and Language Processing*. The author further wishes to acknowledge Dorothea Yantosca for her indispensable design criticism on the graphs and tables which appear in this paper.

References

Thorsten Brants, Ashok C. Popat, Peng Xu, Franz J. Och, and Jeffrey Dean. 2007. Large Language Models in Machine Translation. pages 858–867. Association for Computational Linguistics.

Marc Brysbaert, Boris New, and Emmanuel Keuleers. 2012. Adding part-of-speech information to the SUBTLEX-US word frequencies. Online. https://link.springer.com/content/pdf/10.3758%2Fs13428-012-0190-4.pdf. Accessed online March 10, 2018.

Mark Davies. 2012. Untitled. Online. https://corpus.byu.edu/coca/files/100k_samples.txt. Accessed online March 10, 2018.

Mark Davies. 2018. Word frequency data. Online. https://www.wordfrequency.info/100k_samples.asp. Accessed online March 10, 2018.

Roger Garside, Geoffrey Leech, Michael Bryant, and Nicholas Smith. 2007. UCREL CLAWS7 Tagset. Online. http://ucrel.lancs.ac.uk/claws7tags.html. Accessed online March 10, 2018.

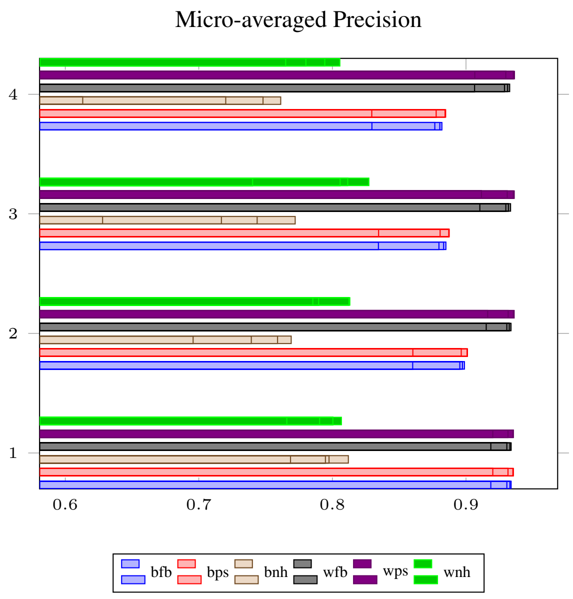
Daniel Jurafsky and James H. Martin. 2017. *Speech and Language Processing*, 3rd edition. Draft of August 28, 2017.

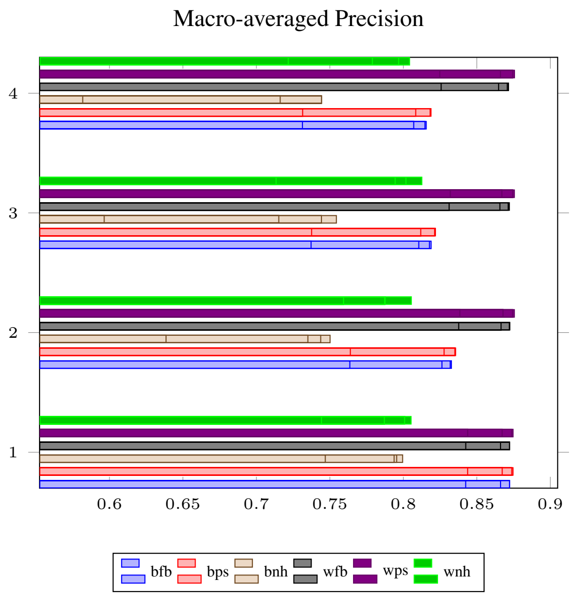
Joakim Nivre, Filip Ginter, Sampo Pyysalo, and Dan Zeman. 2018. Universal POS Tags. Online. http://universaldependencies.org/u/pos/. Accessed online March 4, 2018.

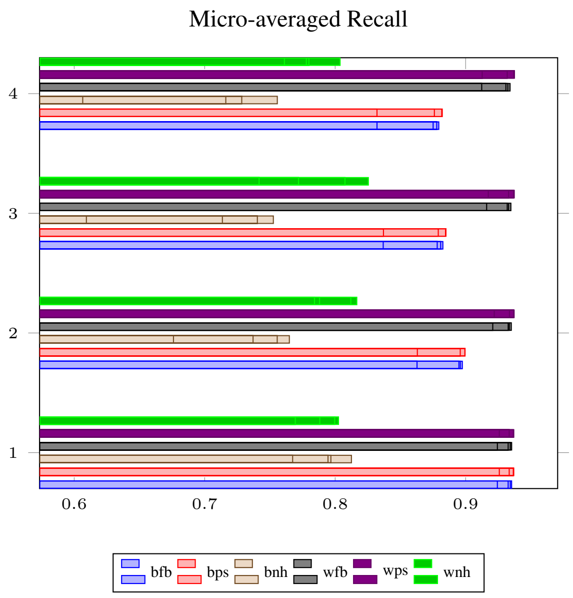
A Supplemental Material

Bar graphs corresponding to the tables given in the body of the paper are given here as a quick visual reference[2]. Those interested in the raw data from the experiment that informed the tables and graphs may contact the author via e-mail.

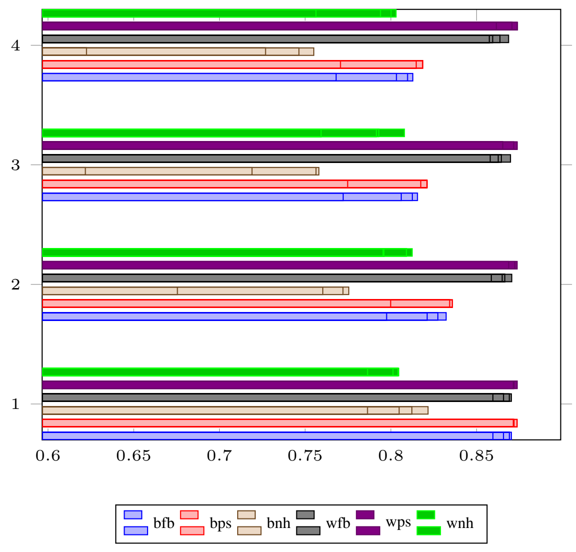
[2](b|w)xx = (backoff|weighted), x(fb|ps|nh) = (fallback|preseed|no heuristic)



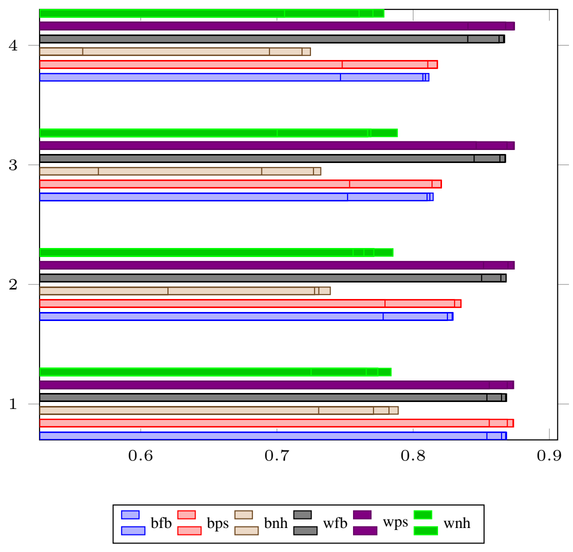




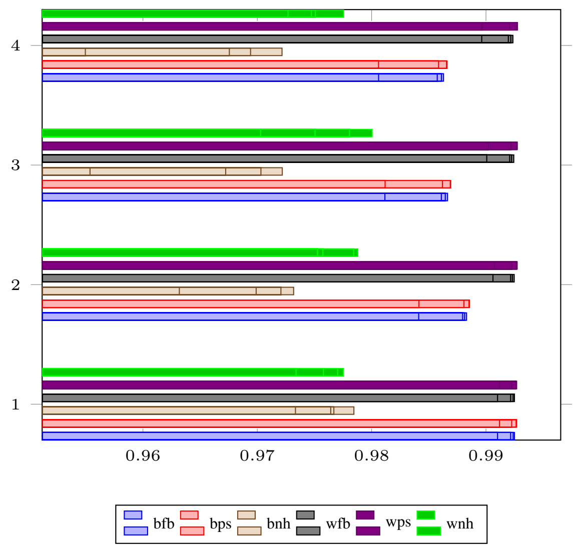
Macro-averaged Recall



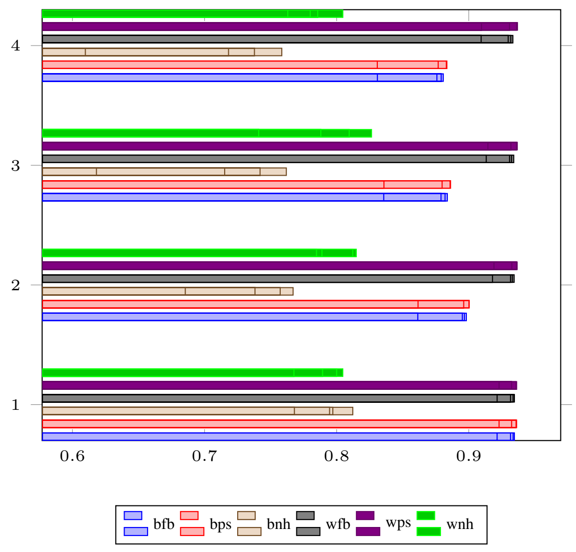
Macro-averaged F_1 Measure



Accuracy



Micro-averaged F_1 Measure


bfb
bps
bnh
wfb
wps
wnh