Chinese Radicals in NLP Tasks

Alex Fandrianto
afandria@stanford.edu

Anand Natarajan
anandn@stanford.edu

Hanzhi Zhu
hanzhiz@stanford.edu

December 7, 2012

1 Introduction

The Chinese writing system uses a set of tens of thousands of characters, each of which represents a syllable, and usually a morpheme, in the Chinese language. (In Chinese, almost all morphemes are monosyllabic). Although the number of distinct characters is very large, they have internal structure, and all of them are made out of a few hundred common graphic components called radicals. Moreover, the vast majority of Chinese characters are so-called phonosemantic compounds: they consist of two parts, a semantic part which is a single radical that indicates something about the meaning of the character, and a phonetic part which gives a hint as to its pronunciation. For example, the character "妈 (mā)," meaning "mother" is composed of a semantic component "女," meaning "woman," and phonetic component "马," pronounced "mǎ." C hinese lexicographers have developed standardized sets of radicals, which are used to organize characters in dictionaries, as well as in the Unihan database which is a part of Unicode.

Normally when processing Chinese, this internal structure is ignored. However, we believe that radicals carry useful information for many NLP tasks. In this project, we investigated how using radicals affected performance on three tasks: language modeling, part-of-speech tagging, and word segmentation.

2 Language Modeling

2.1 Theory

Most modern language models are n-gram models, i.e. they assume that the probability of a word is dependent only on the previous (n-1) words. These probabilities are estimated by counting the occurrences of all n-grams in a corpus, and then smoothing the counts to handle unobserved n-grams. Usually, smoothing involves backing off to probabilities from k-gram models for all k $<$ n; we back off by dropping words from our conditioning context starting from the least recent word. In Chinese, since text is usually not word segmented, it is reasonable to treat each character as a separate word for the purposes of language modeling, which is what we did in our experiments.

One can generalize n-gram models to include other features of the previous words in the following manner: if f_i is a feature of the i-th word w_i, then we replace the probability $p(w_0 \mid w_{-1}w_{-2}\dots w_{-n+1})$ with

$$p(w_0 \mid w_{-1}f_{-1}\dots w_{-n+1}f_{-n+1})p(f_{-1}\dots f_{-n+1} \mid w_{-1}\dots w_{-n+1}).$$

If the feature is a deterministic function of the word, e.g. the semantic radical, then the last term in the in the probability can be ignored. The remaining term is a conditional probability that can be estimated by counting and smoothing, as with an n-gram model.

Since the feature we added is deterministic, the model as defined above should give identical results to an n-gram model. However, the features can make a difference when performing interpolation or backoff, when a word w_i has not been previously observed but the feature value f_i has been. Thus, having a good backoff model is expected to be important for getting good performance from the added features. Moreover,

unlike the case of n-grams, it is no longer obvious in what order to drop conditioning factors when backing off. In general one could have a backoff graph, where each node is a probability model that backs off to some function (e.g. max, min, mean) of the node below it. The space of possible backoff graphs grows with the factorial of the number of conditioning variables, so it can be very large even for a trigram model with one additional feature. Searching this space effectively was a major challenge in this study.

2.2 Experiments

In our particular case, we used the SRILM factored language model package [1], which supports a generalization of n-gram language models that includes models of the type described above. We used simple word-based bigram and trigram models as baselines. To these, we added as a feature the semantic radical, as determined from the Unihan database. For the sake of comparison, we tried models that used the part-of-speech tag of the word containing the given character instead of the semantic radical. For bigram models, we explored the backoff space by hand, but for trigram models, this proved infeasible so we used a genetic algorithm-based tool provided with the package. We experimented with both Kneser-Ney and Witten-Bell smoothing for backoff graph nodes. The dataset used was taken from the Chinese Treebank, with the training set consisting of around 18,000 sentences, and the dev and test sets consisting of around 350 sentences. Performance was measured by the perplexity of the language model.

2.3 Results and Discussion

We consistently observed a decrease in performance (i.e. increased for perplexity) with radicals as compared to baseline, in contrast to a significant increase in performance with part-of-speech tags. This held true for both the bigram and trigram models, and over many runs of the genetic algorithm with different random seeds. For instance, for one configuration, a bigram model achieved a perplexity of 166.3 on the dev set and bigrams with part-of-speech tags achieved a perplexity of 116.8, whereas bigrams with radicals had a perplexity of 193. Differences of similar magnitude were obtained for many language model configurations. Such a robust trend indicates that radicals are likely not actually very useful features in language modeling. Another way of stating this is that the current character is not strongly correlated to the radicals of the preceding characters. This conclusion is consistent with results from part-of-speech tagging experiments (see section 3), where we found that radicals of previous word are not a helpful feature, although the radical of the current word is. To further investigate this claim, we performed the following very simple experiment: we created a language model to predict the radical of the current character given the radical of the previous character, and compared the perplexity of this model to one with uniform probabilities. The uniform model had a perplexity of 100.2, while the radical model had a perplexity of 80.84. This suggests that radicals of successive characters are indeed not very strongly correlated, which implies that the previous radical is not helpful in predicting the next character. We believe that the performance *decreases* when radicals are added because during backoff, probability mass must be removed from the full model and given to the models with dropped conditioning factors. This will decrease the probability the model assigns to sentences without unknown words, and thus increase the perplexity on those sentences. Since the vast majority of "words" (i.e. characters) in our dev set had appeared in our training set, this resulted in overall perplexities increasing.

3 Part-of-speech Tagging

As mentioned in the introduction, one of the radicals in the character usually carries semantic information about the character. From examining a few examples, it becomes clear that certain semantic radicals occur more often in words with a certain part of speech. For instance, the radical "扌" (a reduced form of the character "手" meaning "hand") has a connotation of applying force to something, and occurs almost exclusively in verbs. We thus have strong reason to believe that radicals should be a good feature for part-of-speech tagging tasks.

The Stanford POS-Tagger, a maxent classifier built on top of Stanford' s Core NLP library, was used to investigate the addition of radical features to POS-tagging [5]. Features are specified in configuration files for training. Chinese radicals were added as part of the specified FeatureExtractors through the WordShapeClassifier and RadicalMap.

There were several ways of selecting the radical features; each Chinese word is composed of potentially multiple characters. The main problem is that while there may be a variable number of characters, only a fixed number of specified features per word can be extracted. radicalFirst takes the first character' s radical and returns that as the feature. radicalConcatenated takes each radical from every character and concatenates them. Upon obtaining preliminary results from these methods, radical3 and radicalLast were implemented. The former computes 3 features, the first radical, second radical, and third radical, returning the empty string if there is no corresponding character in the Chinese word. The latter returns the radical of the last character in the word.

Once these specifications were complete, various combinations of features were used to train the POS-Tagger. The baselines compared were 'unigram', 'simple', and 'normal'. The first feature set consists of the current word being tagged. The second is a trigram model. The last was the default 'nodistsim' model, which consists of many features. In general, the addition of radicals improves tagging accuracy, especially for unknown words. Improvements in performance are less visible when the feature set is more complex. Table 1 shows the accuracy data for the various models when run on dev and test. Figures 1 and 2 illustrate model performance when applied to known words and unknown words.

	dev	test	dev_unknown	test_unknown
unigram	63.451107	62.35015	0	0
u+radicalFirst	74.094708	76.735764	21.338156	34.172662
u+radicals	79.313884	79.420579	11.392405	7.194245
u+radical3	83.638763	82.854645	56.238698	50.719424
u+radicalLast	72.819235	75.774226	25.135624	38.848921
simple	92.420466	94.218282	54.611212	71.582734
s+radicalFirst	93.329424	94.343157	60.940325	71.582734
s+radicals	93.2268	94.368132	59.674503	71.223022
s+radical3	94.69286	94.68032	75.406872	79.496403
normal	95.748424	95.37962	84.629295	85.971223
n+radicalFirst	95.865709	95.442058	85.714286	88.129496
n+radicals	95.689782	95.454545	84.448463	87.05036
n+radical3	95.89503	95.404595	85.714286	84.532374
radical3	83.638763	82.742258	56.962025	51.079137
radical3Bigram	86.761472	86.426074	57.685353	67.625899

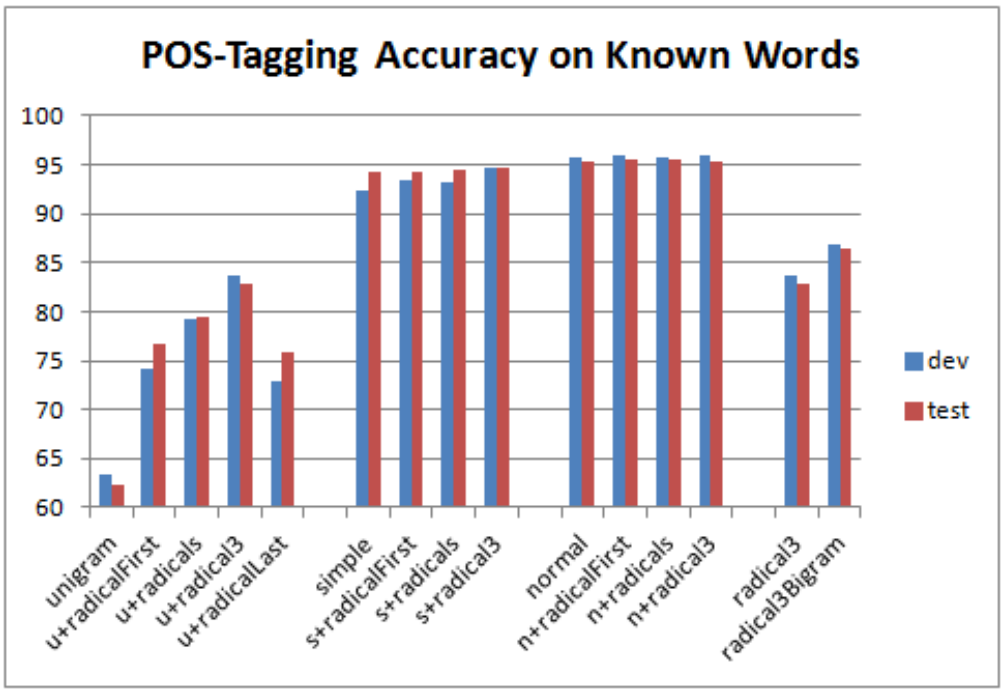

Figure 1: POS-Tagging Accuracy on Known Words. The addition of radical information significantly improves the tagging performance on known words. More mileage is gained when the original feature set is small, like the unigram model, while improvements are less visible when the feature set is large, like in the normal model. radical3 gives the biggest performance gain and taken alone, performs comparably to unigram+radical3.

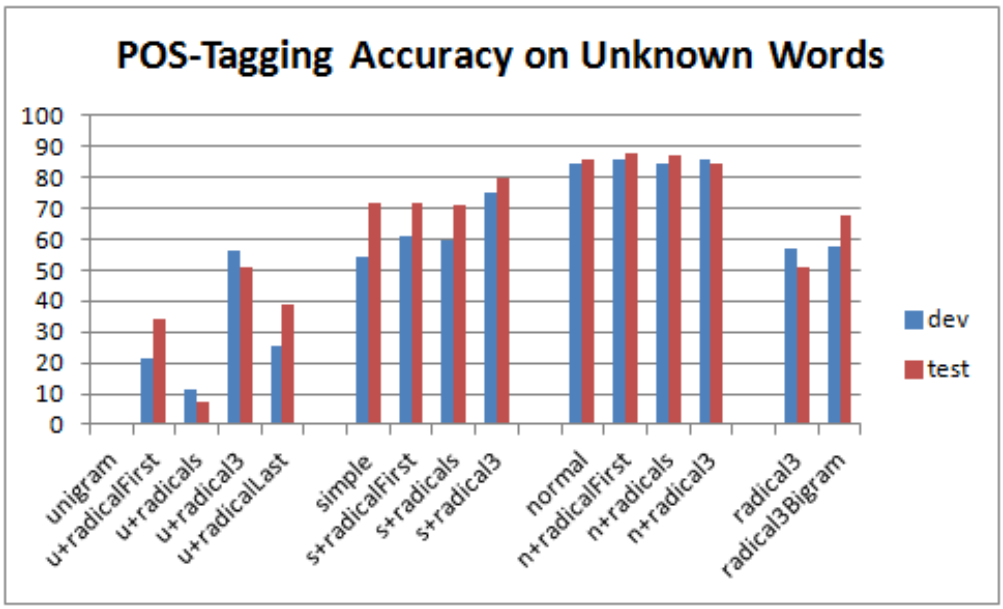

Figure 2: POS-Tagging Accuracy on Unknown Words. The addition of radical information greatly the tagging performance on unknown words. Unlike the unknown word, its radicals have been seen during training. The hefty gains in performance imply that these radicals do in fact correlate with POS.

The data demonstrates that radical3, taking the first 3 radicals as separate features, provides the biggest improvement to the POS-tagging models. When broken down into radicalFirst and radicalLast, the performance gains are not as large, implying that radicalLast and radicalFirst do not solely determine POS. It seems that every radical in the Chinese word can potentially affect the final POS tag.

The feature radicals, the concatenation of the primary radicals in the Chinese word, does not generalize to unknown words very well. On the plus side, an exact match of radical ordering does seem to add a strong performance gain for known words.

While not fully explored, the use of radical3Bigrams did not add much beyond radical3. It is possible that part of speech doesn't depend much on previous words' radicals.

Some limitations of the Stanford POS-Tagger were discovered during the course of training. It is not possible to only train on a single feature. That is, the unigram model below was not fully optimized. Further, it was not possible to use radical4 or radical5 as features because the Maxent Tagger would fail during the optimization process. It is likely that not enough Chinese words are 4 or 5 characters long, but information is still lost if these later character' s radicals are unused.

4 Word Segmentation

The processing of written Chinese presents an additional challenge compared to that of most other languages due to the fact that separate Chinese words are not separated from each other by a word divider (i.e. a whitespace). A Chinese word is composed of a positive number of characters, averaging around two. Thus, the task of Chinese word segmentation becomes necessary to suitably model Chinese.

We view this as a label tagging problem which can be modeled by a maximum entropy or a conditional random field (CRF) framework, the latter which seems to be more commonly adopted [2, 3]. Using CRF for word segmentation, we have two possible tags. For each character we observe, we can either label it as 'B', if it begins new word, or 'I', if it is a continuation of a word (i.e. if it is part of the same word as the previous character).

4.1 Features

We consider using radicals as features for our CRF system. Let us define C_x to be the character x positions after the current one, and R_x to be the radical of C_x. Thus, R_0 is the radical of the current character, R_{-1} is that of the previous character, R_1 that of the next one, and so forth. We implement three features groups, which we will call RadicalUnigram, RadicalBigram, and RadicalTrigram. RadicalUnigram uses R_0, R_{-1}, R_1 as features, namely the unigram. RadicalBigram uses as features the bigram concatenations of adjacent characters' radicals: $R_{-2}R_{-1}$, $R_{-1}R_0$, and R_0R_1. Similarly, RadicalBigram uses the trigram concatenations of three adjacent character's radicals, beginning with the trigram starting at R_{-3} up until the trigram starting at R_0. Our intuition behind choosing which n-grams to use in relation to the current character is as follows. The task of classifying C_0 as B or I is equivalent to deciding whether there is a word boundary between C_{-1} and C_0. The features relevant to this decision should thus be n-grams which include information about the character on either side of this boundary: the n-grams should contain R_{-1} and/or R_0. For example, given the sentence "布朗一行于今晚离沪赴广州", a bigram containing the radicals of 今晚 should intuitively have no relation with whether 行 begins a new word or not.

4.2 Experiments

For our experiment, we started with the most recent release of the Stanford NLP Group's Word Segmenter [4]. We used the same dev and test files as Language Modeling, but we start with a reduced train set that contains around 1600 sentences. We run the CRF on test without adding the radical feature groups, and then iteratively add each n-gram feature group. Without the radical features, the CRF achieved an F score of 0.951. RadicalUnigram did not affect this score, RadicalBigram increased it by 0.1% to 0.952, and RadicalTrigram reduced it to 0.950.

Given the slight increase in performance when using Radical Bigram, we train again our CRF using the full train set as used by Language Modeling. However, without the radical features, we obtain an F measure of 0.981 whereas with them, the F measure reduces slightly to 0.979.

Using the smaller train set, we notice that the out-of-vocabulary (OOV) recall rate has a slightly larger improvement than the overall F-score from running without radical features to running with RadicalBigram, shown below. This was coupled with the fact that using the smaller train set, the rate of unknown (OOV) words occurring was relatively high (16.3%) compared to using the larger train set (3.5%). Intuitively, since training over the radicals generalizes over the space of feature values, the CRF performs better when seeing new character n-grams since the chance that it has seen the radicals of these characters is much higher.

It seems that RadicalBigram performs well with foreign proper names, which tend to get radicals which are most common in such foreign transliterations. With the RadicalBigram feature group, we correctly segmented 理查德 (Richard) and 吉尔吉斯 (Kyrgyz), both tokens of which do not appear in the smaller train set. These two tokens are not correctly segmented without radical features turned on.

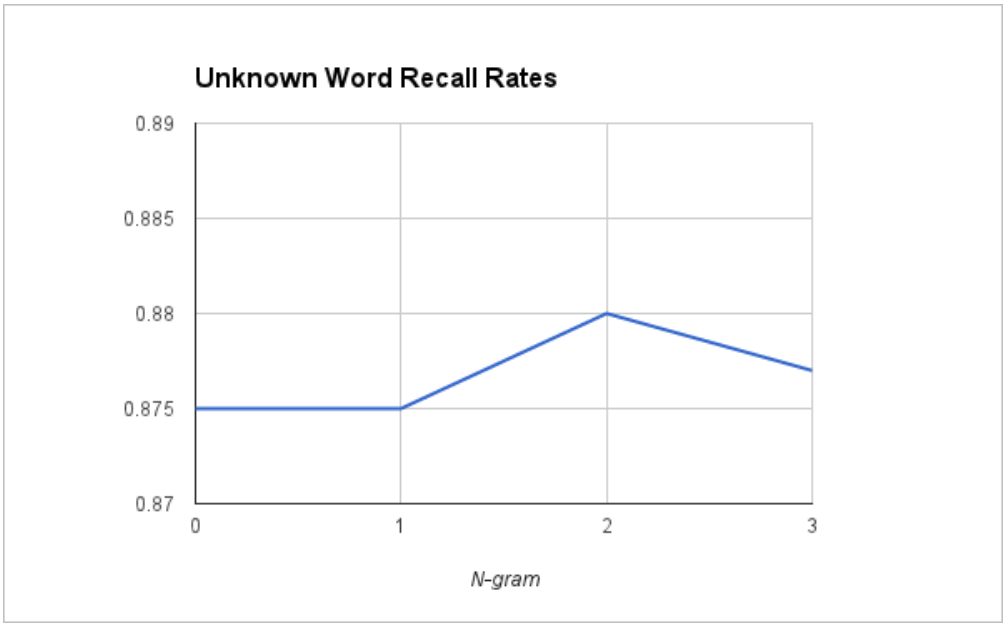

Figure 3: OOV Recall Rates. The rate of OOV recall trained over the smaller dataset for each radical n-gram feature.

Thus, it seems that RadicalBigram only helps improve performance on OOV words, and even then only slightly. When there are fewer OOV words such as when training over a large dataset, the radical features are essentially effectless. It seems that in practice, the high performance of existing CRF word segmenters for Chinese make it so that the additional features will not have a significant impact on their accuracy.

5 Conclusions and Future Work

For future work, we plan to study radicals beyond the main semantic radical, especially for POS tagging. Also, we would like to explore the differences between Simplified and Traditional Chinese for these NLP tasks, since character simplification can be modeled as dropping radicals from characters. We are especially interested in seeing whether simplification dropped radicals in a information-theoretically "optimal" way.

6 Acknowledgements

We would like to thank Mengqiu Wang for his guidance with this project.

References

[1] K. Kirchhoff, J. Bilmes, and K. Duh, *Factored Language Models Tutorial*, (2008), `http://ssli.ee.washington.edu/people/duh/papers/flm-manual.pdf`.

[2] `http://nlp.stanford.edu/pubs/sighan2005.pdf`.

[3] `http://bcmi.sjtu.edu.cn/~zhaohai/pubs/CSB-SIGHAN5_20071015-rev.pdf`.

[4] `http://nlp.stanford.edu/software/segmenter.shtml`.

[5] `http://nlp.stanford.edu/software/tagger.shtml`.